

```

CCCCCCCCCCCCC 000000000 888888888888 RRRRRRRRRRR RRRRRRRRRRRRRR TTTTTTTTTTTTT TTT
CCCCCCCCCCCCC 000000000 888888888888 RRRRRRRRRRRRRR TTTTTTTTTTTTTTTTT TTT
CCCCCCCCCCCCC 000000000 888888888888 RRRRRRRRRRRRRR TTTTTTTTTTTTTTTTT TTT
CCC 000 000 888 888 RRR RRR TTT TTT LLL
CCC 000 000 888 888 RRR RRR TTT TTT LLL
CCC 000 000 888 888 RRR RRR TTT TTT LLL
CCC 000 000 888 888 RRR RRR TTT TTT LLL
CCC 000 000 888 888 RRR RRR TTT TTT LLL
CCC 000 000 888 888 RRR RRR TTT TTT LLL
CCC 000 000 888 888 RRR RRR TTT TTT LLL
CCC 000 000 888888888888 RRRRRRRRRRRRRR TTT TTT LLL
CCC 000 000 888888888888 RRRRRRRRRRRRRR TTT TTT LLL
CCC 000 000 888888888888 RRRRRRRRRRRRRR TTT TTT LLL
CCC 000 000 888 888 RRR RRR TTT TTT LLL
CCC 000 000 888 888 RRR RRR TTT TTT LLL
CCC 000 000 888 888 RRR RRR TTT TTT LLL
CCC 000 000 888 888 RRR RRR TTT TTT LLL
CCC 000 000 888 888 RRR RRR TTT TTT LLL
CCC 000 000 888 888 RRR RRR TTT TTT LLL
CCC 000 000 888 888 RRR RRR TTT TTT LLL
CCCCCCCCCCCCC 000000000 888888888888 RRR RRR RRR TTT LLLLLLLLLLLLLLLLL
CCCCCCCCCCCCC 000000000 888888888888 RRR RRR RRR TTT LLLLLLLLLLLLLLLLL
CCCCCCCCCCCCC 000000000 888888888888 RRR RRR RRR TTT LLLLLLLLLLLLLLLLL

```

[illegible]

```
0001 0 *TITLE 'COBSACCECV - ACCEPT Conversion routines'
0002 0 MODULE COBSACCECV ( IDENT = '1-001' ) = ! File: COBACCECV.B32 EDIT:LGB1001
0003 0
0004 0
0005 1 BEGIN
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *
0028 1 *****
0029 1
0030 1
0031 1 **
0032 1 FACILITY: COBOL SUPPORT
0033 1
0034 1 ABSTRACT:
0035 1
0036 1 Supports the COBOL ACCEPT statement.
0037 1
0038 1 ENVIRONMENT: VAX-11 User Mode
0039 1
0040 1 AUTHOR: Linda Baillie, CREATION DATE: 7-FEB-84
0041 1
0042 1 MODIFIED BY:
0043 1
0044 1 1-001 - Original. LGB 7-FEB-84
0045 1 --
```

COBSACCECV  
1-001

COBSACCECV - ACCEPT Conversion routines

N 12  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 1

```

47 0046 1 |
48 0047 1 | PROLOGUE FILE
49 0048 1 |
50 0049 1 | REQUIRE 'RTLIN:COBPROLOG' ; | Switches, Psects, Include
51 1566 1 | | files
52 1567 1 |
53 1568 1 | LINKAGES:
54 1569 1 |
55 1570 1 | LINKAGE
56 1571 1 |     JSB_678 = JSB
57 1572 1 |         (REGISTER = 6, REGISTER = 7, REGISTER = 8):
58 1573 1 |         NOPRESERVE (2, 3, 4, 5, 6, 7, 8)
59 1574 1 |         NOTUSED (9, 10, 11);
60 1575 1 |
61 1576 1 |     JSB_6789 = JSB
62 1577 1 |         (REGISTER = 6, REGISTER = 7, REGISTER = 8, REGISTER = 9):
63 1578 1 |         NOPRESERVE (2, 3, 4, 5, 6, 7, 8, 9)
64 1579 1 |         NOTUSED (10, 11);
65 1580 1 |
66 1581 1 | TABLE OF CONTENTS:
67 1582 1 |
68 1583 1 | FORWARD ROUTINE
69 1584 1 |     COB$$ACC_CONVERT, | Conversion routine
70 1585 1 |     COB$$NUMERIC_CONV, | Convert to numeric text strings
71 1586 1 |     COB$$COMP_CONV, | Convert to Word, Longword,
72 1587 1 | | Quadword and Packed strings
73 1588 1 |     COB$$FLOAT_CONV, | Convert to Floating and Double
74 1589 1 | | Floating Point strings
75 1590 1 |     COB$$STRIP_BLANKS_SIGN, | Strip blanks and sign from
76 1591 1 | | input numeric string
77 1592 1 |     COB$$ZERO_FILL : NOVALUE, | Initialize STRING_DEST with
78 1593 1 | | zeroes
79 1594 1 |     COB$$VERIFY_FL_RANGE, | Check that input for Floating
80 1595 1 | | Point data items is within range
81 1596 1 |     COB$$SCAN_INPUT ; | Scan input data
82 1597 1 |
83 1598 1 |
84 1599 1 | EQUATED SYMBOLS
85 1600 1 |
86 1601 1 | LITERAL
87 1602 1 |     V_DEC_PT = 64 ; | Bit flag for 'DECIMAL POINT
88 1603 1 | | IS COMMA'
89 1604 1 |
90 1605 1 | EXTERNAL REFERENCES:
91 1606 1 |
92 1607 1 | EXTERNAL ROUTINE
93 1608 1 |
94 1609 1 |     COB$CVTIL_R8: JSB_678, | Convert CIT to long
95 1610 1 |     COB$CVTIP_R9: JSB_6789, | Convert CIT to packed
96 1611 1 |     COB$CVTIQ_R8: JSB_678, | Convert CIT to quad
97 1612 1 |     COB$CVTIW_R8: JSB_678, | Convert CIT to word
98 1613 1 |     COB$CVTTI_R8: JSB_678, | Convert text to CIT
99 1614 1 |     LIB$STOP : NOVALUE, | Signals fatal error
100 1615 1 |     STR$GET1_DX, | Allocate a string
101 1616 1 |     STR$DUPL_CHAR, | Duplicate character n times
102 1617 1 |     STR$FREE1_DX, | Deallocate a string
103 1618 1 |     STR$COPY_R, | Copy a string by ref
```

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines

B 13  
15-Sep-1984 23:49:06 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:10:22 [COBRTL.SRC]COBACCECV.B32;1

Page 3  
(2)

```
: 104      1619 1      COB$$FREE_STRINGS,  
: 105      1620 1      OTSSCVT_I_F,  
: 106      1621 1      OTSSCVT_I_D ;  
: 107      1622 1  
: 108      1623 1 EXTERNAL LITERAL  
: 109      1624 1      COB$_INVARG ;
```

```
! Free local strings  
! Convert Text to Floating point  
! Convert Text to Double Fl point  
  
! Invalid Argument(s)
```

```
111 1625 1 %SBTTL 'COB$ACC_CONVERT - Conversion'
112 1626 1 GLOBAL ROUTINE COB$ACC_CONVERT (STRING_DEST : REF $STR$DESCRIPTOR,
113 1627 1                                     | Final destination for input chars
114 1628 1                                     | Enhancement flag
115 1629 1                                     | Input default value
116 1630 1                                     | Contains input characters
117 1631 1                                     | # of input characters
118 1632 1                                     | =1 if DEFAULT was used
119 1633 1                                     | =1 if sign should be included
120 1634 1                                     |
121 1635 1                                     |
122 1636 1                                     |
123 1637 1                                     |
124 1638 1 ++
125 1639 1 FUNCTIONAL DESCRIPTION:
126 1640 1 Convert TEXT input string to specified VAX COBOL data type.
127 1641 1 This routine selects the appropriate routine to convert the specified
128 1642 1 data type.
129 1643 1
130 1644 1 FORMAL PARAMETERS:
131 1645 1
132 1646 1
133 1647 1 STRING_DEST.mt.ds Address of descriptor to receive the read input.
134 1648 1
135 1649 1 FLAGS.rlu.v Screen enhancement flag;
136 1650 1
137 1651 1 DEFAULT.rt.dx Default source moved to destination descriptor
138 1652 1 (STRING_DEST) in the event of null input.
139 1653 1
140 1654 1 PUT_HERE.rt.dx Buffer to hold input characters.
141 1655 1
142 1656 1 CHARS_READ.rlu.v Number of characters accepted as input.
143 1657 1
144 1658 1 YES_DEFAULT.rlu.v Flag = 1 if DEFAULT used because of null input.
145 1659 1
146 1660 1 YES_SIGN.rlu.v Flag = 1 if sign should be included in COMP or COMP3
147 1661 1 data type.
148 1662 1
149 1663 1 IMPLICIT INPUTS:
150 1664 1
151 1665 1 NONE
152 1666 1
153 1667 1 IMPLICIT OUTPUTS:
154 1668 1
155 1669 1 NONE
156 1670 1
157 1671 1 ROUTINE VALUE:
158 1672 1
159 1673 1 1 - Conversion Success
160 1674 1 0 - Conversion Failure
161 1675 1
162 1676 1 SIDE EFFECTS:
163 1677 1
164 1678 1 Signals COB$_INVARG if the syntax of the number is wrong.
165 1679 1 --
166 1680 1
167 1681 2 BEGIN
```

```
168 1682 2 LOCAL
169 1683 2 ERROR : INITIAL (0) ; = 1, Conversion error
170 1684 2 PUT_HERE_BUF : REF VECTOR [1100, BYTE], : Temp for special case
171 1685 2 CONV_OK : INITIAL (0) ; : Conversion check
172 1686 2 : : Return Status
173 1687 2
174 1688 2
175 1689 2 :+ This validity check does not apply to TEXT, FLOATING or
176 1690 2 DOUBLE FLOATING POINT data types.
177 1691 2
178 1692 2 IF ( .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_T AND
179 1693 2 .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_F AND
180 1694 2 .STRING_DEST [DSC$B_DTYPE] NEQ DSC$K_DTYPE_D )
181 1695 2 THEN
182 1696 2 BEGIN : Begin special case
183 1697 2 :+
184 1698 2 Looking for two special case conversion errors not caught
185 1699 2 in other conversion routines.
186 1700 2 Check for invalid value '-9' and '0000-1234'
187 1701 2 ('9-' found in COBS$NUMERIC_CONV).
188 1702 2
189 1703 2
190 1704 2 IF .YES_DEFAULT : Use local buffer for
191 1705 2 THEN : convenience
192 1706 2 PUT_HERE_BUF = .DEFAULT [DSC$A_POINTER]
193 1707 2 ELSE
194 1708 2 PUT_HERE_BUF = .PUT_HERE [DSC$A_POINTER] ;
195 1709 2
196 1710 2 INCR X FROM 0 TO .CHARS_READ - 1 DO
197 1711 2 BEGIN
198 1712 2 :+
199 1713 2 Work through characters one at a time.
200 1714 2
201 1715 2 SELECTONE .PUT_HERE_BUF [X] OF
202 1716 2 SET
203 1717 2 [X'C'0' TO X'C'9'] :
204 1718 2 0 ; : Legal
205 1719 2 [X'C''] :
206 1720 2 0 ; : Legal
207 1721 2
208 1722 2 [X'C'-' , X'C'+'] :
209 1723 2 IF .X NEQ 0 AND .X NEQ .CHARS_READ - 1
210 1724 2 THEN
211 1725 2 :+
212 1726 2 Looking for 0000-1234 case - a sign in middle
213 1727 2 of digits. Sign should be first or last
214 1728 2 character. Note: bbbb-1234 is legal.
215 1729 2
216 1730 2
217 1731 2 IF (( .PUT_HERE_BUF [.X-1] GEQ X'C'0' AND
218 1732 2 .PUT_HERE_BUF [.X-1] LEQ X'C'9' ) AND
219 1733 2 ( .PUT_HERE_BUF [.X+1] GEQ X'C'0' AND
220 1734 2 .PUT_HERE_BUF [.X+1] LEQ X'C'9' ))
221 1735 2 THEN
222 1736 2 BEGIN
223 1737 2 ERROR = 1 ;
224 1738 2 EXITLOOP ;
```

```
END ;
[XC'.' XC','] :
BEGIN
IF .X NEQ .CHARS_READ - 1
THEN
+
Looking for .-9 case - a sign separating a
decimal point and digit. -.9 and .9- are
acceptable. (9-. caught by COB$NUMERIC_CONV)
IF .PUT_HERE_BUF [.X+1] EQL XC'-' OR
.PUT_HERE_BUF [.X+1] EQL XC' '
THEN
BEGIN
ERROR = 1 ;
EXITLOOP ;
END ;
END ;
[ OTHERWISE ]:
0;
! Let conversion routines
! handle other errors
TES ;
END ;
! End special case
IF .ERROR
THEN
CONV_OK = 0
! Return failure status
ELSE
+
Continue with rest of conversion check. Call appropriate
routine (determined by data type).
CASE .STRING_DEST [DSC$B_DTYPE] FROM DSC$K_DTYPE_WU TO
DSC$K_DTYPE_P OF
SET
[DSC$K_DTYPE_WU, ! Numeric string
DSC$K_DTYPE_NL, DSC$K_DTYPE_NR,
DSC$K_DTYPE_NLO, DSC$K_DTYPE_NRO] :
BEGIN
CONV_OK = COB$NUMERIC_CONV (( IF .YES_DEFAULT
THEN .DEFAULT
ELSE .PUT_HERE ),
.STRING_DEST, .CHARS_READ, .FLAGS ) ;
END ;
[DSC$K_DTYPE_W, DSC$K_DTYPE_WU, ! Word
DSC$K_DTYPE_L, DSC$K_DTYPE_LU, ! Longword
DSC$K_DTYPE_Q, DSC$K_DTYPE_QU, ! Quadword
DSC$K_DTYPE_P] :
! Packed Decimal
BEGIN
CONV_OK = COB$COMP_CONV ( .STRING_DEST,
```



```
282 1796 3
283 1797
284 1798
285 1799
286 1800
287 1801
288 1802
289 1803
290 1804
291 1805
292 1806
293 1807
294 1808
295 1809
296 1810
297 1811
298 1812
299 1813
300 1814
301 1815
302 1816
303 1817
304 1818
305 1819
306 1820
307 1821
308 1822
309 1823
310 1824
311 1825
312 1826
313 1827
314 1828
315 1829
316 1830
317 1831
318 1832
319 1833
320 1834
321 1835
322 1836
323 1837
324 1838
325 1839
326 1840
327 1841
328 1842
329 1843
330 1844
331 1845
332 1846
333 1847
334 1848
335 1849
336 1850
337 1851
338 1852 2

      .FLAGS, .DEFAULT, .PUT_HERE,
      .CHARS_READ, .YES_DEFAULT,
      .YES_SIGN ) ;

      END ;

[DSC$K_DTYPE_F, DSC$K_DTYPE_D] :      ! Floating and Double
                                      ! Floating Point
      BEGIN
      +
      | Since DEFAULT is Read-only, copy it to PUT_HERE if
      | it was used, just in case it is necessary to WRITE
      | a DOT to override a COMMA in routine COB$$FLOAT_CONV.
      -
      IF .YES_DEFAULT
      THEN
          CH$MOVE ( .CHARS_READ, .DEFAULT [DSC$A_POINTER],
                    .PUT_HERE [DSC$A_POINTER] ) ;

          CONV_OK = COB$$FLOAT_CONV ( .STRING_DEST, .FLAGS,
                                       .PUT_HERE, .CHARS_READ ) ;
      END ;

[DSC$K_DTYPE_T] :      ! Text
      +
      | Copy ACCEPTed data to STRING_DEST. If more chars
      | are ACCEPTed than STRING_DEST[DSC$W_LENGTH] can
      | handle, accept only the leftmost characters and
      | ignore the extra characters.
      | Use STR$COPY because it BLANK fills.
      -
      BEGIN
          LOCAL
              COPY_NUM ;

          IF .CHARS_READ LSS .STRING_DEST[DSC$W_LENGTH]
          THEN
              COPY_NUM = .CHARS_READ
          ELSE
              COPY_NUM = .STRING_DEST[DSC$W_LENGTH] ;

          STR$COPY_R ( .STRING_DEST, COPY_NUM,
                      (IF .YES_DEFAULT
                       THEN .DEFAULT [DSC$A_POINTER]
                       ELSE .PUT_HERE [DSC$A_POINTER] ) ) ;

          CONV_OK = 1 ;      ! no need to conv TEXT
      END ;

[INRANGE, OUTRANGE] :

      LIB$STOP ( COB$_INVARG ) ;

      TES ;

      RETURN .CONV_OK ;
```

COBSACCECV  
1-001

COBSACCECV - ACCEPT Conversion routines  
COB\$\$\$ACC\_CONVERT - Conversion

G 13

15-Sep-1984 23:49:06

14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742

[COBRTL.SRC]COBACCECV.B32;1

Page 8

(3)

; 339

1853 1 END ;

! End of COB\$\$\$ACC\_CONVERT

.TITLE COBSACCECV COBSACCECV - ACCEPT Conversion routines

.IDENT \1-001\

.EXTRN COB\$CVTIL\_R8, COB\$CVTIP\_R9

.EXTRN COB\$CVTIQ\_R8, COB\$CVTIW\_R8

.EXTRN COB\$CVTTI\_R8, LIB\$STOP

.EXTRN STR\$GET1\_DX, STR\$DUPL\_CHAR

.EXTRN STR\$FREET\_DX, STR\$COPY\_R

.EXTRN COB\$\$FREE\_STRINGS

.EXTRN OT\$SCVT\_T\_F, OT\$SCVT\_T\_D

.EXTRN COB\$\_INVARG

.PSECT \_COB\$CODE, NOWRT, SHR, PIC, 2

.ENTRY COB\$\$\$ACC\_CONVERT, Save R2,R3,R4,R5,R6,R7

SUBL2 #4, SP

CLRL ERROR

CLRL CONV\_OK

MOVL STRING\_DEST, R6

CMPB 2(R6), #14

BEQL 1\$

CMPB 2(R6), #10

BEQL 1\$

CMPB 2(R6), #11

BNEQ 2\$

BRW 12\$

BLBC YES\_DEFAULT, 3\$

MOVL DEFAULT, R0

BRB 4\$

MOVL PUT\_HERE, R0

MOVL 4(R0), PUT\_HERE\_BUF

MNEGL #1, X

BRB 11\$

MOVZBL (X)[PUT\_HERE\_BUF], R2

CMPB R2, #48

BLSSU 6\$

CMPB R2, #57

BLEQU 11\$

CMPB R2, #32

BEQL 11\$

CMPB R2, #43

BEQL 7\$

CMPB R2, #45

BNEQ 8\$

TSTL X

BEQL 11\$

SUBL3 #1, CHARS\_READ, R2

CPL X, R2

BEQL 11\$

CMPB -1(X)[PUT\_HERE\_BUF], #48

BLSSU 11\$

CMPB -1(X)[PUT\_HERE\_BUF], #57

BGTRU 11\$

SE		00FC	00000			
		04	C2	00002		
		53	D4	00005		
		57	D4	00007		
56	04	AC	D0	00009		
0E	02	A6	91	0000D		
		0A	13	00011		
0A	02	A6	91	00013		
		04	13	00017		
0B	02	A6	91	00019		
		03	12	0001D	1\$:	
		008C	31	0001F		
06	18	AC	E9	00022	2\$:	
50	0C	AC	D0	00026		
		04	11	0002A		
50	10	AC	D0	0002C	3\$:	
50	04	A0	D0	00030	4\$:	
51		01	CE	00034		
		70	11	00037		
52		6140	9A	00039	5\$:	
30		52	91	0003D		
		05	1F	00040		
39		52	91	00042		
		62	1B	00045		
20		52	91	00047	6\$:	
		5D	13	0004A		
2B		52	91	0004C		
		05	13	0004F		
2D		52	91	00051		
		2C	12	00054		
		51	D5	00056	7\$:	
		4F	13	00058		
52	14	AC	01	C3	0005A	
52		51	D1	0005F		
		45	13	00062		
30	FF	A140	91	00064		
		3E	1F	00069		
39	FF	A140	91	0006B		
		37	1A	00070		

COBSACCECV  
1-001

COBSACCECV - ACCEPT Conversion routines  
COB\$\$\$ACC\_CONVERT - Conversion

H 13  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 9  
(3)

		30	01	A140	91	00072	CMPB	1(X)[PUT_HERE_BUF], #48	1733	
		30			1F	00077	BLSSU	11\$		
		39	01	A140	91	00079	CMPB	1(X)[PUT_HERE_BUF], #57	1734	
				29	1A	0007E	BGTRU	11\$		
				22	11	00080	BRB	10\$	1737	
		2C		52	91	00082	8\$: CMPB	R2, #44	1741	
				05	13	00085	BEQL	9\$		
		2E		52	91	00087	CMPB	R2, #46		
				1D	12	0008A	BNEQ	11\$		
	52	14	AC	01	C3	0008C	9\$: SUBL3	#1, CHARS_READ, R2	1743	
			52	51	D1	00091	CMPL	X, R2		
				13	13	00094	BEQL	11\$		
			2D	01	A140	91	00096	CMPB	1(X)[PUT_HERE_BUF], #45	1750
				07	13	00098	BEQL	10\$		
			20	01	A140	91	0009D	CMPB	1(X)[PUT_HERE_BUF], #32	1751
				05	12	000A2	BNEQ	11\$		
		53		01	D0	000A4	10\$: MOVL	#1, ERROR	1754	
				05	11	000A7	BRB	12\$	1753	
	88		51	14	AC	F2	11\$: AOBLSS	CHARS_READ, X, 5\$	1710	
			04		53	E9	12\$: BLBC	ERROR, 13\$	1766	
					57	D4	CLRL	CONV_OK	1768	
					38	11	BRB	16\$		
		12	03	02	A6	8F	13\$: CASEB	2(R6), #3, #18	1774	
0026	0050	0050		0050		000BA	14\$: .WORD	20\$-14\$,-		
0065	0050	0050		0050		000C2		20\$-14\$,-		
0088	0026	0026		0065		000CA		20\$-14\$,-		
0035	0035	0035		0035		000D2		15\$-14\$,-		
	0050	0026		0035		000DA		20\$-14\$,-		
								20\$-14\$,-		
								20\$-14\$,-		
								21\$-14\$,-		
								21\$-14\$,-		
								15\$-14\$,-		
								15\$-14\$,-		
								25\$-14\$,-		
								17\$-14\$,-		
								17\$-14\$,-		
								17\$-14\$,-		
								17\$-14\$,-		
								15\$-14\$,-		
								20\$-14\$		
								#COB\$ INVARG	1848	
	00000000G	00		00000000G	8F	DD	15\$: PUSHL	#1, LIB\$STOP		
					01	FB	CALLS	24\$		
					54	11	BRB			
				08	AC	DD	16\$: PUSHL	FLAGS	1786	
				14	AC	DD	17\$: PUSHL	CHARS_READ		
					56	DD	PUSHL	R6		
		05		18	AC	E9	BLBC	YES DEFAULT, 18\$	1783	
				0C	AC	DD	PUSHL	DEFAULT	1784	
					03	11	BRB	19\$		
				10	AC	DD	18\$: PUSHL	PUT_HERE	1785	
	0000V	CF			04	FB	19\$: CALLS	#4, -COB\$\$\$NUMERIC_CONV	1783	
					36	11	BRB	23\$		
		7E		18	AC	7D	20\$: MOVQ	YES DEFAULT, -(SP)	1797	
		7E		10	AC	7D	MOVQ	PUT_HERE, -(SP)	1796	
		7E		08	AC	7D	MOVQ	FLAGS, -(SP)		

COBSACCECV  
1-001

COBSACCECV - ACCEPT Conversion routines  
COB\$ACC\_CONVERT - Conversion

I 13  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 10  
(3)

		0000V	CF		56	DD	00116		PUSHL	R6		1795
					07	FB	00118		CALLS	#7, COB\$\$\$COMP_CONV		
					21	11	0011D		BRB	23\$		
			OF	18	AC	E9	0011F	21\$:	BLBC	YES DEFAULT, 22\$		1810
			51	0C	AC	D0	00123		MOVL	DEFAULT, R1		1812
			50	10	AC	D0	00127		MOVL	PUT HERE, R0		1813
	04	B0			14	AC	28	00128	MOVC3	CHARS_READ, @4(R1), @4(R0)		
		04	B1	14	AC	7D	00132	22\$:	MOVQ	PUT HERE, -(SP)		1816
			7E	10	AC	DD	00136		PUSHL	FLAGS		1815
				08	AC	DD	00139		PUSHL	R6		
		0000V	CF		56	DD	0013B		CALLS	#4, COB\$\$\$FLOAT_CONV		
			57		04	FB	0013B		MOVL	R0, CONV_OK		
					50	D0	00140	23\$:	BRB	30\$		1774
					31	11	00143	24\$:	CMPZV	#0, #16, (R6), CHARS_READ		1833
14	AC				00	ED	00145	25\$:	BLEQ	26\$		
			10		06	15	00148		MOVL	CHARS_READ, COPY_NUM		1835
			6E	14	AC	D0	0014D		BRB	27\$		
					03	11	00151		MOVZWL	(R6), COPY_NUM		1837
			6E		66	3C	00153	26\$:	BLBC	YES DEFAULT, 28\$		1840
			06	18	AC	E9	00156	27\$:	MOVL	DEFAULT, R0		1841
			50	0C	AC	D0	0015A		BRB	29\$		
					04	11	0015E		MOVL	PUT HERE, R0		1842
			50	10	AC	D0	00160	28\$:	PUSHL	4(R0)		
				04	A0	DD	00164	29\$:	PUSHAB	COPY_NUM		1839
				04	AE	9F	00167		PUSHL	R6		
		00000000G	00		56	DD	0016A		CALLS	#3, STR\$COPY_R		
			57		03	FB	0016C		MOVL	#1, CONV_OK		1843
			50		01	D0	00173		MOVL	CONV_OK, R0		1852
					57	D0	00176	30\$:	RET			1853
					04	00179						

; Routine Size: 378 bytes, Routine Base: \_COB\$CODE + 0000

; 340 1854 1

```
1855 1 %SBTTL 'COB$$NUMERIC_CONV - Convert to numeric string'
1856 1 ROUTINE COB$$NUMERIC_CONV ( Scan a number and divide it up
1857 1 ARG DESC : REF BLOCK [8, BYTE], The number to scan
1858 1 STRING_DEST : REF BLOCK [12, BYTE], Final resting place of input
1859 1 CHARS_READ, Number of characters read
1860 1 FLAGS-) = Needed for DECIMAL POINT IS COMMA
1861 1
1862 1 **
1863 1 FUNCTIONAL DESCRIPTION:
1864 1
1865 1 Convert a TEXT input string to the appropriate VAX COBOL Text
1866 1 Numeric data type. Pull off decimal point, pull off sign then
1867 1 place it in the correct position for internal representation.
1868 1 Do nothing about errors in this routine, return control to COB$ACC_SCR
1869 1 via COB$$ACC_CONVERT.
1870 1
1871 1 FORMAL PARAMETERS:
1872 1
1873 1 ARG_DESC.rt.dx The number to parse
1874 1
1875 1 STRING_DEST.mt.ds Address of descriptor to receive the read input.
1876 1
1877 1 CHARS_READ.rlu.v Number of characters accepted as input.
1878 1
1879 1 FLAGS.rlu.v Screen enhancement flag;
1880 1
1881 1 IMPLICIT INPUTS:
1882 1
1883 1 NONE
1884 1
1885 1 IMPLICIT OUTPUTS:
1886 1
1887 1 NONE
1888 1
1889 1 ROUTINE VALUE:
1890 1
1891 1 1 = Conversion Success
1892 1 0 = Conversion Failure
1893 1
1894 1 SIDE EFFECTS:
1895 1
1896 1 Signals COB$_INVARG if the syntax of the number is wrong,
1897 1 --
1898 1
1899 2 BEGIN
1900 2
1901 2 LOCAL
1902 2 SIGN_VAL : BYTE, Holds + or - sign
1903 2 BUF_DESC : BLOCK [8, BYTE] VOLATILE, Temporary buffer
1904 2 SIGN_SEEN : INITIAL (0), 1 = we have seen a + or -
1905 2 DIGIT_SEEN : INITIAL (0), 1 = we have seen at least one digit
1906 2 DOT_SEEN : INITIAL (0), 1 = we have seen a decimal point
1907 2 BLANKS_SEEN : INITIAL (0), 1 = we have seen trailing blanks
1908 2 ZERO_SEEN : INITIAL (0), 1 = zero seen
1909 2 PUTTER : INITIAL (0), Counts position in the output buffer
1910 2 BUF : REF VECTOR [1100, BYTE], Addresses result
1911 2 ARG : REF VECTOR [1100, BYTE], Addresses source
```

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$NUMERIC\_CONV - Convert to numeric string

K 13  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 12  
(4)

```

399 1912 2 ARG_LEN, | Length of the source
400 1913 2 NUM_DIGITS : INITIAL (0), | Number of digits in ARG_DESC
401 1914 2 LEFT_DEC : INITIAL (0), | Number of digits to left of dec pt
402 1915 2 RIGHT_DEC : INITIAL (0), | Number of digits to right of dec pt
403 1916 2 LEADING_ZEROES: INITIAL (0), | Counter of leading zeroes
404 1917 2 OK_LEFT; | Correct number of digits allowed
405 1918 2 | to left of decimal point
406 1919 2
407 1920 2 BIND
408 1921 2 ZERO = UPLIT ('0');
409 1922 2
410 1923 2 | +
411 1924 2 | Enable a handler to free the local string in case of an error.
412 1925 2 | -
413 1926 2 ENABLE
414 1927 2 COB$$FREE_STRINGS (BUF_DESC);
415 1928 2
416 1929 2 | +
417 1930 2 | Allocate enough space to hold the digits. It is convenient to
418 1931 2 | allocate before scanning, so we may allocate a little too much,
419 1932 2 | but the space will be freed before we return.
420 1933 2 | -
421 1934 2 BUF_DESC [DSC$W_LENGTH] = 0;
422 1935 2 BUF_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
423 1936 2 BUF_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
424 1937 2 BUF_DESC [DSC$A_POINTER] = 0;
425 1938 2 ARG_LEN = .ARG_DESC [DSC$W_LENGTH];
426 1939 2 STR$GET1_DX (%REF (.ARG_LEN + 20), BUF_DESC);
427 1940 2 | +
428 1941 2 | Set pointers.
429 1942 2 | -
430 1943 2 BUF = .BUF_DESC [DSC$A_POINTER];
431 1944 2 ARG = .ARG_DESC [DSC$A_POINTER];
432 1945 2 SIGN_VAL = %C'+';
433 1946 2
434 1947 2 | +
435 1948 2 | Scan the input number, put result in BUF.
436 1949 2 | -
437 1950 2
438 1951 2 IF NOT ( COB$$SCAN_INPUT ( .ARG_DESC, .CHARS_READ, .FLAGS, BUF_DESC,
439 1952 2 LEFT_DEC, NUM_DIGITS, SIGN_VAL, PUTTER, LEADING_ZEROES, SIGN_SEEN,
440 1953 2 DIGIT_SEEN, DOT_SEEN, ZERO_SEEN, BLANKS_SEEN ) )
441 1954 2 THEN
442 1955 2 RETURN 0 ;
443 1956 2
444 1957 2 | +
445 1958 2 | Now ensure that a number of the form 1.0000 has the trailing
446 1959 2 | zeroes stripped off.
447 1960 2 | -
448 1961 2 IF .DOT_SEEN
449 1962 2 THEN
450 1963 2 BEGIN
451 1964 2 LOCAL
452 1965 2 X;
453 1966 2
454 1967 2 X = .PUTTER - 1;
455 1968 2 IF .X NEQ 0
```

COBSACCECV  
1-001

COBSACCECV - ACCEPT Conversion routines  
COB\$\$NUMERIC\_CONV - Convert to numeric string

L 13  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 13  
(4)

```

456      1969 3      THEN
457      1970 4      BEGIN
458      1971 4      WHILE .BUF[.X] EQL %C'0' DO
459      1972 5      BEGIN
460      1973 5      NUM_DIGITS = .NUM_DIGITS - 1;
461      1974 5      IF .X EQL 0
462      1975 5      THEN
463      1976 6      BEGIN
464      1977 6      DIGIT_SEEN = 0;
465      1978 6      DOT_SEEN = 0;
466      1979 6      EXITLOOP;
467      1980 5      END;
468      1981 5      X = .X - 1;
469      1982 4      END;
470      1983 3      END;
471      1984 2      END;
472      1985 2
473      1986 2
474      1987 2
475      1988 2
476      1989 2
477      1990 2
478      1991 2
479      1992 2
480      1993 2
481      1994 2
482      1995 2
483      1996 2
484      1997 3
485      1998 3
486      1999 3
487      2000 3
488      2001 4
489      2002 3
490      2003 3
491      2004 3
492      2005 3
493      2006 3
494      2007 3
495      2008 3
496      2009 3
497      2010 3
498      2011 3
499      2012 3
500      2013 3
501      2014 3
502      2015 3
503      2016 3
504      2017 3
505      2018 3
506      2019 3
507      2020 2
508      2021 2
509      2022 2
510      2023 2
511      2024 3
512      2025 3

      NOTE: Call to COB$$ZERO_FILL was originally done here, however because
      VAX RPG wants an illegal string returned in its original state,
      it was necessary to be more selective about where and when to
      call COB$$ZERO_FILL.

      If there are no digits, or only leading zeros, take the number to
      be zero. Don't be too gullible, however.

      IF ( NOT .DIGIT_SEEN )
      THEN
      BEGIN
      IF (.SIGN_SEEN OR .DOT_SEEN OR .BLANKS_SEEN) AND (.ZERO_SEEN EQL 0)
      THEN RETURN 0 ;

      Fill STRING_DEST with zeroes

      COB$$ZERO_FILL ( .STRING_DEST ) ;
      RETURN 1 ;
      END

      Validate size of entered data, left and right of decimal point.
      If everything is OK, copy the input string to STRING_DEST with
      the sign set up correctly.
      Return if DEFAULT is being converted (there will never be a decimal point
      in the DEFAULT parameter)

      ELSE
      BEGIN
      LOCAL
      DEST_PTR ,
      DIGITS_IN_STRING ,

      ! Throw out trailing zeroes after dec pt.
      ! Get here if all digits after decimal point are zeroes, and zeroes are the only digits in the string (eg. .0000)
      ! Pointer where result will go in destination
      ! Number of digits in destination string
```

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$NUMERIC\_CONV - Convert to numeric string

M 13

15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 14  
(4)

```

513      2026      3      DEST_LENGTH ;
514      2027      3      LENGTH_DIFF ;
515      2028      3
516      2029      3
517      2030      3
518      2031      3      LITERAL
519      2032      3      POSOP = 16,
520      2033      3      NEGOP = 25,
521      2034      3      POSZEROP = 123,
522      2035      3      NEGZEROP = 125;
523      2036      3
524      2037      3
525      2038      4      IF NOT (.DOT SEEN)
526      2039      3      THEN LEFT_DEC = .NUM_DIGITS ;
527      2040      3      RIGHT_DEC = .NUM_DIGITS - .LEFT_DEC ;
528      2041      3
529      2042      3      DEST_LENGTH = .STRING_DEST [DSC$W_LENGTH];
530      2043      3
531      2044      3      SELECTONE .STRING_DEST [DSC$B_CLASS] OF
532      2045      3      SET
533      2046      3
534      2047      3      [ DSC$K_CLASS_S ] :
535      2048      3
536      2049      4      BEGIN
537      2050      4
538      2051      4      !+
539      2052      4      ! If a decimal point was typed in, all the digits after it
540      2053      4      ! MUST be zeroes.
541      2054      4      !-
542      2055      4
543      2056      4      IF .RIGHT_DEC GTR 0
544      2057      4      THEN
545      2058      4          INCR I FROM (.PUTTER - .RIGHT_DEC) TO .PUTTER DO
546      2059      4              IF .BUF[I] NEQ XC'0'
547      2060      4                  THEN RETURN 0;
548      2061      4
549      2062      4      !+
550      2063      4      ! If the number of digits typed in is less than the number of
551      2064      4      ! digits in the destination string, then a pointer must be
552      2065      4      ! set up here so that the typed in digits get moved to the
553      2066      4      ! correct place in the destination.
554      2067      4      !-
555      2068      4
556      2069      4      DIGITS_IN_STRING =
557      2070      5          (-IF (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_NL) OR
558      2071      6              (.STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_NR)
559      2072      5              THEN .DEST_LENGTH - 1
560      2073      4              ELSE .DEST_LENGTH );
561      2074      4      IF .LEFT_DEC GTR .DIGITS_IN_STRING
562      2075      4      THEN RETURN 0;
563      2076      4
564      2077      4      STR$DUPL CHAR (.STRING_DEST, DEST_LENGTH, ZERO); ! Zero the destination
565      2078      4      LENGTH_DIFF = .DIGITS_IN_STRING - .LEFT_DEC;
566      2079      4      IF .LENGTH_DIFF GTR 0
567      2080      4      THEN
568      2081      4          DEST_PTR = .STRING_DEST [DSC$A_POINTER] + .LENGTH_DIFF
569      2082      4      ELSE
```



COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$NUMERIC\_CONV - Convert to numeric string

N 13  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 15  
(4)

```

: 570      2083 4          DEST_PTR = .STRING_DEST [DSC$a_POINTER];
: 571      2084 4
: 572      2085 3          END ;
: 573      2086 3
: 574      2087 3          [ DSC$K_CLASS_SD ] :
: 575      2088 3
: 576      2089 4          BEGIN
: 577      2090 4          | +
: 578      2091 4          | All code for "P" data types are in lowercase.
: 579      2092 4          | -
: 580      2093 4
: 581      2094 4          LOCAL
: 582      2095 4          ok_right,
: 583      2096 4          LENGTH_DIFF2;
: 584      2097 4          | Difference between number of digits
: 585      2098 4          | to the right of the decimal point
: 586      2099 4          | in the typed in number and the dest
: 587      2100 4          dest_length = .string_dest[dsc$b_digits];
: 588      2101 4
: 589      2102 4          | +
: 590      2103 4          | This is checking for the P Picture of 99PP.
: 591      2104 4          | If the scale is positive and the number of digits in the
: 592      2105 4          | number equal the scale factor, then simply copy the digits
: 593      2106 4          | in BUF to the destination descriptor.
: 594      2107 4          | -
: 595      2108 4
: 596      2109 4          if .string_dest[dsc$b_scale] gtr 0
: 597      2110 4          then
: 598      2111 5              begin
: 599      2112 5                  local
: 600      2113 5                  tot_digits,
: 601      2114 5                  diff;
: 602      2115 5
: 603      2116 5                  if ((.right_dec gtr 0) or (.num_digits gtr (.string_dest[dsc$b_digits] + .string_dest[ds
: 604      2117 6                  then
: 605      2118 5                      | number too large
: 606      2119 5                      return 0;
: 607      2120 5                      | re-prompt - error
: 608      2121 5
: 609      2122 5                  if .num_digits leq .string_dest[dsc$b_scale]
: 610      2123 5                  then
: 611      2124 5
: 612      2125 5                  | +
: 613      2126 5                  | Zero out the destination field using the digits as the proper
: 614      2127 5                  | number of zero fill characters, rather than using the length
: 615      2128 5                  | as found in the descriptor, since class SD is a special case.
: 616      2129 6                  | -
: 617      2130 6                  begin
: 618      2131 6                  str$dupl_char (.string_dest, dest_length, zero);
: 619      2132 5                  return 1;
: 620      2133 5                  | answer is zero
: 621      2134 5                  end ;
: 622      2135 5
: 623      2136 6                  if .leading_zeroes neq 0
: 624      2137 6                  then
: 625      2138 6                      begin
: 626      2139 6                          diff = (.string_dest[dsc$b_digits]+.string_dest[dsc$b_scale]) - .num_digits;
                          dest_ptr = .string_dest[dsc$a_pointer]+.diff;
```

```

627      2140 6
628      2141 6
629      2142 5
630      2143 6
631      2144 6
632      2145 6
633      2146 6
634      2147 6
635      2148 6
636      2149 6
637      2150 7
638      2151 7
639      2152 7
640      2153 7
641      2154 7
642      2155 6
643      2156 6
644      2157 6
645      2158 6
646      2159 5
647      2160 5
648      2161 5
649      2162 4
650      2163 5
651      2164 5
652      2165 5
653      2166 5
654      2167 5
655      2168 5
656      2169 5
657      2170 5
658      2171 5
659      2172 5
660      2173 6
661      2174 6
662      2175 6
663      2176 6
664      2177 6
665      2178 6
666      2179 6
667      2180 6
668      2181 6
669      2182 6
670      2183 6
671      2184 6
672      2185 6
673      2186 6
674      2187 6
675      2188 6
676      2189 6
677      2190 6
678      2191 6
679      2192 6
680      2193 6
681      2194 6
682      2195 6
683      2196 6

      end
    else
      begin
        diff = .num_digits - .string_dest[dsc$b_scale];
        if .diff eq[ .string_dest[dsc$b_digits]
        then
          dest_ptr = .string_dest[dsc$a_pointer]
        else
          begin
            tot_digits = (.string_dest[dsc$b_digits] + .string_dest[dsc$b_scale]) - .num_dig
            dest_ptr = .string_dest[dsc$a_pointer] + .tot_digits;
          end;
          num_digits = .diff;
        end;
      end
    else
      begin
        OK_LEFT = .STRING_DEST [DSC$B_DIGITS] + .STRING_DEST [DSC$B_SCALE] ;
        if .ok_left lss 0
        then
          +
          | Here we have a P Picture field of type PP99.
          | We know this when OK_LEFT is less than zero.
          | It requires some special casing.
          -
          begin
            local
              diff,
              diff2,
              ptr,
              buf_ptr;

            if .left_dec gtr 0
            then
              ! error no '.' entered
              ! ring bell and reprompt
              return 0;

            ok_left = 0;
            ok_right = abs(.string_dest[dsc$b_scale]);
            if .right_dec gtr .ok_right
            then
              return 0;

          +
          | This handles case where the number of digits
          | entered is less than the absolute value of the
          | scale factor, meaning that the number returned
          | would have to be zero. The first part of the
          | if statement takes care of the case where the

```

```

684      2197 6      ! number of digits entered equals the number of
685      2198 6      ! digits expected taking into account if the absolute
686      2199 6      ! value of the scale factor is equal to the number
687      2200 6      ! of digits entered to the right of the decimal point
688      2201 6      ! thereby giving us a result of zero again.
689      2202 6
690      2203 6
691      2204 7
692      2205 6      diff = (abs(.string_dest[dsc$b_scale]) -
693      2206 7      .string_dest[dsc$b_digits]); ! Number of Placeholders in picture
694      2207 6      if ((.right_dec eql .string_dest[dsc$b_digits]) and
695      2208 7      (.right_dec eql .diff)) OR
696      2209 6      (.right_dec leq .diff)
697      2210 7      then
698      2211 7      begin
699      2212 7      str$dupl_char (.string_dest, dest_length, zero);
700      2213 6      return 1; ! done - answer is zero
701      2214 6      end ;
702      2215 6
703      2216 6      buf_ptr = .buf_desc [dsc$a_pointer] + .diff;
704      2217 6      diff = .num_digits - .diff; ! Number of digits minus placeholders
705      2218 6      ch$move (.diff, .buf_ptr, .buf_desc[dsc$a_pointer]);
706      2219 6      diff2 = .string_dest[dsc$b_digits] - .diff; ! Number of digits to zero fill
707      2220 6      ptr = .buf_desc[dsc$a_pointer] + .diff;
708      2221 6      incr i from 0 to .diff2 - 1 do
709      2222 6      ch$move (1, zero, .ptr + .i);
710      2223 6
711      2224 6      dest_ptr = .string_dest[dsc$a_pointer];
712      2225 6      num_digits = .string_dest[dsc$b_digits] ; ! should only reflect number of digits
713      2226 6
714      2227 5      end
715      2228 6      else
716      2229 6      begin
717      2230 6      LENGTH_DIFF = .OK_LEFT - .LEFT_DEC;
718      2231 6      LENGTH_DIFF2 = (.STRING_DEST [DSC$b_DIGITS] - .OK_LEFT) - .RIGHT_DEC;
719      2232 7      IF ( .LENGTH_DIFF LSS 0) OR ( .LENGTH_DIFF2 LSS 0)
720      2233 6      THEN
721      2234 6      RETURN 0 ; ! Data entered too big
722      2235 6
723      2236 6      !+
724      2237 6      ! If the number of digits to the left of the decimal
725      2238 6      ! point of the number typed in is less than what
726      2239 6      ! should be in the destination string, then a pointer
727      2240 6      ! must be set up here so that the typed in digits get
728      2241 6      ! moved to the correct place in the destination.
729      2242 6      !-
730      2243 6      IF .LENGTH_DIFF GTR 0
731      2244 6      THEN
732      2245 6      DEST_PTR = .STRING_DEST [DSC$a_POINTER] + .LENGTH_DIFF
733      2246 6      ELSE
734      2247 6      DEST_PTR = .STRING_DEST [DSC$a_POINTER];
735      2248 6
736      2249 6      !+
737      2250 6      ! If the number of digits to the right of the decimal
738      2251 6      ! point of the number typed in is less than what
739      2252 6      ! should be in the destination string, then the typed
740      2253 6      ! in number must be padded with trailing zeroes so that
```

```
741 2254 6      ! sign placement can be done correctly.
742 2255 6
743 2256 6      IF .LENGTH_DIFF2 GTR 0
744 2257 6      THEN
745 2258 7          BEGIN
746 2259 7
747 2260 7          LOCAL
748 2261 7              PTR;
749 2262 7              ! Pointer into input buffer
750 2263 7              ! past the digits that were typed in
751 2264 7          PTR = .BUF_DESC [DSC$A_POINTER] + .NUM_DIGITS - 1;
752 2265 7          NUM_DIGITS = .NUM_DIGITS + .LENGTH_DIFF2;
753 2266 7
754 2267 7          DO
755 2268 8              BEGIN
756 2269 8
757 2270 8              LOCAL
758 2271 8                  PTR2;
759 2272 8                  ! Loop pointer
760 2273 8
761 2274 8                  PTR2 = .PTR + .LENGTH_DIFF2;
762 2275 8                  CH$MOVE (1, ZERO, .PTR2);
763 2276 8                  LENGTH_DIFF2 = .LENGTH_DIFF2 - 1;
764 2277 8
765 2278 7              END
766 2279 7              UNTIL .LENGTH_DIFF2 EQL 0;
767 2280 6          END;
768 2281 6
769 2282 5      end;
770 2283 4      end;
771 2284 3      END ;
772 2285 3
773 2286 3      [ OTHERWISE ] :
774 2287 3
775 2288 3          LIB$STOP ( COB$_INVARG ) ;
776 2289 3
777 2290 3      TES ;
778 2291 3
779 2292 3      !+
780 2293 3      ! Fill STRING_DEST with zeroes.
781 2294 3      ! If everything is OK, copy the input string to STRING_DEST with
782 2295 3      ! the sign set up correctly.
783 2296 3      !-
784 2297 3
785 2298 3      COB$$ZERO_FILL ( .STRING_DEST ) ;
786 2299 3
787 2300 3      CASE .STRING_DEST [DSC$B_DTYPE] FROM DSC$K_DTYPE_NU TO DSC$K_DTYPE_NRO
788 2301 3      OF
789 2302 3      SET
790 2303 3
791 2304 3          [DSC$K_DTYPE_NU]:
792 2305 3              ! Numeric unsigned
793 2306 3              !+
794 2307 3              ! Simply ignore a sign if it was part of the input
795 2308 3              ! string.
796 2309 3              CH$MOVE (.NUM_DIGITS, .BUF_DESC [DSC$A_POINTER], .DEST_PTR);
797 2310 3
```

```
798 2311 3 [DSC$K_DTYPE_NL]: ! Numeric left separate
799 2312 3
800 2313 4 BEGIN
801 2314 4
802 2315 4 CH$MOVE (1, SIGN_VAL, .STRING_DEST [DSC$A_POINTER]);
803 2316 4 CH$MOVE (.NUM_DIGITS, .BUF_DESC [DSC$A_POINTER], .DEST_PTR + 1);
804 2317 4
805 2318 3 END;
806 2319 3
807 2320 3 [DSC$K_DTYPE_NR]: ! Numeric right separate
808 2321 3
809 2322 4 BEGIN
810 2323 4
811 2324 4 BUF [.NUM_DIGITS] = .SIGN_VAL;
812 2325 4 CH$MOVE (.NUM_DIGITS + 1, .BUF_DESC [DSC$A_POINTER], .DEST_PTR);
813 2326 4
814 2327 3 END;
815 2328 3
816 2329 3 [DSC$K_DTYPE_NLO]: ! Numeric left overpunched
817 2330 3
818 2331 4 BEGIN
819 2332 4 LOCAL
820 2333 4 FIRST_TWO : VECTOR [2, BYTE], ! To be compared with
821 2334 4 ! first two bytes of STRING_DEST
822 2335 4 FIRST_DIGIT : BYTE ; ! First digit and overpunch sign
823 2336 4
824 2337 4 FIRST_TWO [0] = %X'7B' ; ! Positive overpunched 0
825 2338 4 FIRST_TWO [1] = %X'30' ; ! Regular 0
826 2339 4
827 2340 4 !+
828 2341 4 ! First byte of initial state STRING_DEST is always 7B,
829 2342 4 ! therefore have to look at first two bytes.
830 2343 4 !-
831 2344 4
832 2345 4 CH$MOVE (.NUM_DIGITS, .BUF_DESC [DSC$A_POINTER], .DEST_PTR) ;
833 2346 4 IF CH$EQL (2, .STRING_DEST [DSC$A_POINTER], 2, FIRST_TWO )
834 2347 4
835 2348 4 THEN
836 2349 4 FIRST_DIGIT =
837 2350 5 (-IF .SIGN_VAL EQL %C'+'
838 2351 5 THEN POSZEROP
839 2352 5 ELSE NEGZEROP )
840 2353 4
841 2354 4 ELSE
842 2355 4 !+
843 2356 4 ! Special treatment needed when .NUM_DIGITS is less
844 2357 4 ! than total # STRING_DEST can hold.
845 2358 4 ! For example : S99V99 - input 3.00 ->
846 2359 4 ! First two bytes of STRING_DEST now hold '7B' and '33',
847 2360 4 ! but want POSZEROP not POSOP as POSOP will result in wrong
848 2361 4 ! number 33.00.
849 2362 4 !-
850 2363 4 IF .NUM_DIGITS LSS .STRING_DEST [DSC$W_LENGTH]
851 2364 4 THEN
852 2365 5 FIRST_DIGIT =
853 2366 5 (-IF .SIGN_VAL EQL %C'+'
854 2367 5 THEN POSZEROP
854 2367 5 ELSE NEGZEROP )
```

```

855 2368 4 ELSE
856 2369 4 FIRST DIGIT =
857 2370 4
858 2371 4 Add whatever necessary to a regular hex number
859 2372 4 to put it in overpunch format. Have to special
860 2373 4 case zero.
861 2374 4
862 2375 5 ( IF .SIGN_VAL EQL %C'+
863 2376 5 THEN
864 2377 5 IF .BUF [0] EQL %C'0'
865 2378 5 THEN POSZEROP
866 2379 5 ELSE .BUF [0] + POSOP
867 2380 5 ELSE
868 2381 5 IF .BUF [0] EQL %C'0'
869 2382 5 THEN NEGZEROP
870 2383 5 ELSE .BUF [0] + NEGOP );
871 2384 4
872 2385 4 CH$MOVE (1, FIRST_DIGIT, .STRING_DEST [DSC$A_POINTER] ) ;
873 2386 4
874 2387 3 END;
875 2388 3 [DSC$K_DTYPE_NRO]: ! Numeric right overpunched
876 2389 3
877 2390 3 BEGIN
878 2391 4
879 2392 4
880 2393 4 Add whatever necessary to a regular hex number
881 2394 4 to put it in overpunch format. Have to special
882 2395 4 case zero.
883 2396 4
884 2397 4
885 2398 4 IF .BUF [.NUM_DIGITS - 1] EQL %C'0'
886 2399 4 THEN
887 2400 4 BUF [.NUM_DIGITS - 1] =
888 2401 5 ( IF .SIGN_VAL EQL %C'+
889 2402 5 THEN POSZEROP
890 2403 5 ELSE NEGZEROP )
891 2404 4 ELSE
892 2405 4 BUF [.NUM_DIGITS - 1] =
893 2406 5 ( IF .SIGN_VAL EQL %C'+
894 2407 5 THEN .BUF [.NUM_DIGITS - 1] + POSOP
895 2408 4 ELSE .BUF [.NUM_DIGITS - 1] + NEGOP );
896 2409 4
897 2410 4 CH$MOVE (.NUM_DIGITS, .BUF_DESC [DSC$A_POINTER], .DEST_PTR);
898 2411 4
899 2412 3 END;
900 2413 3
901 2414 3 TES;
902 2415 3
903 2416 2 END ;
904 2417 2
905 2418 2
906 2419 2
907 2420 2
908 2421 2 STR$FREE1_DX (BUF_DESC);
909 2422 2 RETURN 1 ;
910 2423 1 END;
! end of COB$$NUMERIC_CONV
```

Address	Op Code	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
---------	---------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

		0C	AE	7C	0009F	CLRQ	DOT_SEEN	1978
			04	11	000A2	BRB	5\$	1976
			50	D7	000A4	DECL	X	1981
			EA	11	000A6	BRB	3\$	1971
	56	08	AC	D0	000A8	MOVL	STRING_DEST, R6	2008
	18	10	AE	E8	000AC	BLBS	DIGIT_SEEN, 8\$	1997
	08	14	AE	E8	000B0	BLBS	SIGN_SEEN, 6\$	2001
	04	0C	AE	E8	000B4	BLBS	DOT_SEEN, 6\$	
	05	04	AE	E9	000B8	BLBC	BLANKS_SEEN, 7\$	
		08	AE	D5	000BC	TSTL	ZERO_SEEN	
			C3	13	000BF	BEQL	1\$	
			56	DD	000C1	PUSHL	R6	2008
	0000V	CF		01	FB	CALLS	#1, COB\$\$\$ZERO_FILL	
				0276	31	BRW	61\$	2009
						BLBS	DOT_SEEN, 9\$	2038
	05	0C	AE	E8	000CB	MOVL	NUM_DIGITS, LEFT_DEC	2039
	28	24	AE	D0	000CF	MOVL	NUM_DIGITS, R5	2040
	55	24	AE	D0	000D4	MOVL	LEFT_DEC, R7	
	57	28	AE	D0	000D8	SUBL3	R7, R5, RIGHT_DEC	
52	55		57	C3	000DC	MOVZWL	(R6), DEST_LENGTH	2042
	2C		66	3C	000E0	MOVZBL	3(R6), R0	2044
	50		03	A6	9A	CMPB	R0, #1	2047
	01		50	91	000E8	BNEQ	17\$	
			54	12	000EB	TSTL	RIGHT_DEC	2056
			52	D5	000ED	BLEQ	12\$	
			14	15	000EF	SUBL3	RIGHT_DEC, PUTTER, R0	2058
50	1C	AE	52	C3	000F1	DECL	I	2059
			50	D7	000F6	BRB	11\$	
			06	11	000F8	CMPB	(I)[BUF], #48	
	30		6049	91	000FA	BNEQ	1\$	
			84	12	000FE	AOBLEQ	PUTTER, I, 10\$	
F5	50	1C	AE	F3	00100	CMPB	2(R6), #16	2070
	10	02	A6	91	00105	BEQL	13\$	
			06	13	00109	CMPB	2(R6), #18	2071
	12	02	A6	91	0010B	BNEQ	14\$	
			07	12	0010F	SUBL3	#1, DEST_LENGTH, DIGITS_IN_STRING	2072
53	2C	AE	01	C3	00111	BRB	15\$	
			04	11	00116	MOVL	DEST_LENGTH, DIGITS_IN_STRING	2073
	53	2C	AE	D0	00118	CMPL	R7, DIGITS_IN_STRING	2074
	53		57	D1	0011C	BGTR	19\$	
			45	14	0011F	PUSHL	R11	2077
			5B	DD	00121	PUSHAB	DEST_LENGTH	
			30	AE	9F	PUSHL	R6	
			56	DD	00126	CALLS	#3, STR\$DUPL_CHAR	
	00000000G	00	03	FB	00128	SUBL2	R7, LENGTH_DIFF	2078
		53	57	C2	0012F	BLEQ	16\$	2079
			07	15	00132	ADDL3	4(R6), LENGTH_DIFF, DEST_PTR	2081
58	53	04	A6	C1	00134	BRB	23\$	
			77	11	00139	MOVL	4(R6), DEST_PTR	2083
	58	04	A6	D0	0013B	BRB	23\$	2044
			71	11	0013F	CMPB	R0, #9	2087
	09		50	91	00141	BEQL	18\$	
			03	13	00144	BRW	39\$	
			011B	31	00146	MOVZBL	9(R6), R4	2100
	54	09	A6	9A	00149	MOVL	R4, DEST_LENGTH	
	2C	AE	54	D0	0014D	MOVAB	4(R6), RTO	2139
	5A	04	A6	9E	00151	CVTBL	8(R6), R8	2109
	58	08	A6	98	00155			



			59	15	00159	BLEQ	24\$			
			52	D5	00158	TSTL	RIGHT_DEC		2117	
			6F	14	0015D	BGTR	26\$			
	50		58	C1	0015F	ADDL3	R8, R4, R0			
		50	55	D1	00163	CMPL	R5, R0			
			66	14	00166	BGTR	26\$			
		58	55	D1	00168	CMPL	R5, R8		2121	
			79	15	0016B	BLEQ	30\$			
		18	AE	D5	0016D	TSTL	LEADING_ZEROES		2134	
			15	13	00170	BEQL	20\$			
		51	09	A6	9A	00172	MOVZBL	9(R6), R1	2138	
		50	08	A6	98	00176	CVTBL	8(R6), R0		
		51		50	C0	0017A	ADDL2	R0, R1		
	50		55	C3	0017D	SUBL3	R5, R1, DIFF			
	58		50	C1	00181	ADDL3	DIFF, (R10), DEST_PTR		2139	
			2B	11	00185	BRB	23\$		2134	
		50	08	A6	98	00187	CVTBL	8(R6), DIFF	2145	
		55		50	C3	0018B	SUBL3	DIFF, R5, DIFF		
50	09	A6	08	00	ED	0018F	CMPL	#0, #8, 9(R6), DIFF	2146	
			05	12	00195	BNEQ	21\$			
		58		6A	D0	00197	MOVL	(R10), DEST_PTR	2148	
			12	11	0019A	BRB	22\$			
		51	09	A6	9A	0019C	MOVZBL	9(R6), R1	2152	
		53	08	A6	98	001A0	CVTBL	8(R6), R3		
		51		53	C0	001A4	ADDL2	R3, R1		
		51		55	C2	001A7	SUBL2	R5, TOT_DIGITS		
	58		51	C1	001AA	ADDL3	TOT_DIGITS, (R10), DEST_PTR		2153	
		24	AE	50	D0	001AE	MOVL	DIFF, NUM_DIGITS	2157	
				72	11	001B2	BRB	34\$	2109	
	50		54	58	C1	001B4	ADDL3	R8, R4, OK_LEFT	2165	
				6E	18	001B8	BGEQ	35\$	2166	
				57	D5	001BA	TSTL	R7	2181	
				10	14	001BC	BGTR	26\$		
			50	D4	001BE	CLRL	OK_LEFT		2185	
		50		58	D0	001C0	MOVL	R8, R0	2186	
			03	18	001C3	BGEQ	25\$			
		50		50	CE	001C5	MNEGL	R0, R0		
		51		50	D0	001C8	MOVL	R0, OK_RIGHT		
		51		52	D1	001CB	CMPL	RIGHT_DEC, OK_RIGHT	2187	
			03	15	001CE	BLEQ	28\$			
			0172	31	001D0	BRW	62\$			
	57		54	C3	001D3	SUBL3	R4, R0, DIFF		2205	
		54		52	D1	001D7	CMPL	RIGHT_DEC, R4	2206	
			05	12	001DA	BNEQ	29\$			
		57		52	D1	001DC	CMPL	RIGHT_DEC, DIFF	2207	
			05	13	001DF	BEQL	30\$			
		57		52	D1	001E1	CMPL	RIGHT_DEC, DIFF	2208	
			11	14	001E4	BGTR	31\$			
			5B	DD	001E6	PUSHL	R11		2211	
			30	AE	9F	001E8	PUSHAB	DEST_LENGTH		
				56	DD	001EB	PUSHL	R6		
		00000000G	00	03	FB	001ED	CALLS	#3, STR\$DUPL_CHAR		
				014A	31	001F4	BRW	61\$	2212	
	50		57	AE	C1	001F7	ADDL3	BUF_DESC+4, DIFF, BUF_PTR	2215	
	57			57	C3	001FC	SUBL3	DIFF, R5, DIFF	2216	
38	8c		60	57	28	00200	MOVC	DIFF, (BUF_PTR), @BUF_DESC+4	2217	
			52	09	A6	9A	00205	MOVZBL	9(R6), DIFF2	2218

50	52	38	57	C2	00209	SUBL2	DIFF, DIFF2	2219	
	57		AE	C1	0020C	ADDL3	BUF_DESC+4, DIFF, PTR	2221	
	51		01	CE	00211	MNEGL	#1, -1		
			04	11	00214	BRB	33\$		
	6140		6B	90	00216	32\$:	MOVB	ZERO, (1)[PTR]	
F8	51		52	F2	0021A	33\$:	AOBLS	DIFF2, I, 32\$	
	58		6A	D0	0021E		MOVL	(R10), DEST_PTR	2223
24	AE	09	A6	9A	00221		MOVZBL	9(R6), NUM_DIGITS	2224
			49	11	00226	34\$:	BRB	40\$	2166
53	50		57	C3	00228	35\$:	SUBL3	R7, OK LEFT, LENGTH_DIFF	2230
	54		50	C2	0022C		SUBL2	OK LEFT, R4	2231
51	54		52	C3	0022F		SUBL3	RIGHT DEC, R4, LENGTH_DIFF2	
			53	D5	00233		TSTL	LENGTH_DIFF	2232
			99	19	00235		BLSS	27\$	
			51	D5	00237		TSTL	LENGTH_DIFF2	
			95	19	00239		BLSS	27\$	
			53	D5	0023B		TSTL	LENGTH_DIFF	2243
58	6A		06	15	0023D		BLEQ	36\$	
	58		53	C1	0023F		ADDL3	LENGTH_DIFF, (R10), DEST_PTR	2245
			03	11	00243		BRB	37\$	
			6A	D0	00245	36\$:	MOVL	(R10), DEST_PTR	2247
			51	D5	00248	37\$:	TSTL	LENGTH_DIFF2	2256
			25	15	0024A		BLEQ	40\$	
50	55	38	AE	C1	0024C		ADDL3	BUF_DESC+4, R5, R0	2264
			50	D7	00251		DECL	PTR	
	24	AE	51	C0	00253		ADDL2	LENGTH_DIFF2, NUM_DIGITS	2265
52	50		51	C1	00257	38\$:	ADDL3	LENGTH_DIFF2, PTR, PTR2	2273
	62		6B	90	0025B		MOVB	ZERO, (PTR2)	2274
			51	D7	0025E		DECL	LENGTH_DIFF2	2275
			F5	12	00260		BNEQ	38\$	2278
			0D	11	00262		BRB	40\$	2044
	00000000G	00	8F	DD	00264	39\$:	PUSHL	#COB\$ INVARG	2288
			01	FB	0026A		CALLS	#1, LIB\$STOP	
	0000V	CF	56	DD	00271	40\$:	PUSHL	R6	2298
		57	01	FB	00273		CALLS	#1, COB\$\$ZERO FILL	
04	OF	24	AE	D0	00278		MOVL	NUM_DIGITS, R7	2309
001A	002A	000D	02	A6	8F	0027C	CASEB	2(R6), #15, #4	2300
			00B1		00281	41\$:	.WORD	59\$-41\$,-	
			007E		00289			42\$-41\$,-	
								44\$-41\$,-	
								43\$-41\$,-	
								54\$-41\$	
			00A4	31	0028B		BRW	59\$	2309
			AE	90	0028E	42\$:	MOVB	SIGN_VAL, @4(R6)	2315
01	A8	38	57	28	00293		MOVC3	R7, @BUF_DESC+4, 1(DEST_PTR)	2316
			62	11	00299		BRB	53\$	2300
	6749		AE	90	0029B	43\$:	MOVB	SIGN_VAL, (R7)[BUF]	2324
	50	20	A7	9E	002A0		MOVAB	1(R7), R0	2325
68	38	BE	50	28	002A4		MOVC3	R0, @BUF_DESC+4, (DEST_PTR)	
			52	11	002A9		BRB	53\$	2300
	30	AE	8F	B0	002AB	44\$:	MOVW	#12411, FIRST_TWO	2337
68	38	BE	57	28	002B1		MOVC3	R7, @BUF_DESC+4, (DEST_PTR)	2345
	30	AE	B6	B1	002B6		CMPW	@4(R6), FIRST_TWO	2346
			07	13	002BB		BEQL	45\$	
57	66	10	00	ED	002BD		CMPZV	#0, #16, (R6), R7	2362
			08	15	002C2		BLEQ	46\$	
	2B	20	AE	91	002C4	45\$:	CMPB	SIGN_VAL, #43	2365

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$NUMERIC\_CONV - Convert to numeric string

K 14  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 25  
(4)

		20	12	002C8	BNEQ	50\$		
		0B	11	002CA	BRB	47\$		
	2B	20	AE	91 002CC	46\$:	CMPB	SIGN_VAL, #43	2375
		13	12	002D0	BNEQ	49\$		
	30		69	91 002D2	CMPB	(BUF), #48		2377
		06	12	002D5	BNEQ	48\$		
	50	7B	8F	9A 002D7	47\$:	MOVZBL	#123, R0	
		19	11	002DB	BRB	52\$		
	50		69	9A 002DD	48\$:	MOVZBL	(BUF), R0	2379
	50		10	C0 002E0	ADDL2	#16, R0		
		11	11	002E3	BRB	52\$		2377
	30		69	91 002E5	49\$:	CMPB	(BUF), #48	2381
		06	12	002E8	BNEQ	51\$		
	50	7D	8F	9A 002EA	50\$:	MOVZBL	#125, R0	
		06	11	002EE	BRB	52\$		
	50		69	9A 002F0	51\$:	MOVZBL	(BUF), R0	2383
	50		19	C0 002F3	ADDL2	#25, R0		
	51		50	90 002F6	52\$:	MOVB	R0, FIRST_DIGIT	2375
04	B6		51	90 002F9	53\$:	MOVB	FIRST_DIGIT, @4(R6)	2385
		38	11	002FD	54\$:	BRB	60\$	2300
	50	FF	A749	9E 002FF	54\$:	MOVAB	-1(R7)[BUF], R0	2398
	30		60	91 00304	CMPB	(R0), #48		
		12	12	00307	BNEQ	56\$		
	2B	20	AE	91 00309	CMPB	SIGN_VAL, #43		2401
		06	12	0030D	BNEQ	55\$		
	51	7B	8F	9A 0030F	MOVZBL	#123, R1		
		1A	11	00313	BRB	58\$		
	51	7D	8F	9A 00315	55\$:	MOVZBL	#125, R1	
		14	11	00319	BRB	58\$		
	2B	20		91 0031B	56\$:	CMPB	SIGN_VAL, #43	2406
		08	12	0031F	BNEQ	57\$		
	51		60	9A 00321	MOVZBL	(R0), R1		2407
	51		10	C0 00324	ADDL2	#16, R1		
		06	11	00327	BRB	58\$		
	51		60	9A 00329	57\$:	MOVZBL	(R0), R1	2408
	51		19	C0 0032C	ADDL2	#25, R1		
	60		51	90 0032F	58\$:	MOVB	R1, (R0)	2406
68	38	BE	57	28 00332	59\$:	MOVCB	R7, @BUF_DESC+4, (DEST_PTR)	2410
			AE	9F 00337	60\$:	PUSHAB	BUF_DESC	2421
00000000G	00		01	FB 0033A	CALLS	#1, STR\$FREE1_DX		
	50		01	D0 00341	61\$:	MOVL	#1, R0	2422
			04	00344	RET			
		50	04	00345	62\$:	CLRL	R0	2423
			04	00347	RET			
			0000	00348	63\$:	.WORD	Save nothing	1899
	50	08	AC	D0 0034A	MOVL	8(AP), R0		
	50	04	A0	D0 0034E	MOVL	4(R0), R0		
		F8	A0	9F 00352	PUSHAD	BUF_DESC		
			01	DD 00355	PUSHL	#1		
			5E	DD 00357	PUSHL	SP		
	7E	04	AC	7D 00359	MOVW	4(AP), -(SP)		
00000000G	00		03	FB 0035D	CALLS	#3, COB\$\$FREE_STRINGS		
			04	00364	RET			

; Routine Size: 869 bytes, Routine Base: \_COB\$CODE + 0180

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$FLOAT\_CONV - Convert to Floating Point

L 14  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 15

```

912 2424 1 %SBTTL 'COB$$FLOAT_CONV - Convert to Floating Point'
913 2425 1 ROUTINE COB$$FLOAT_CONV (STRING_DEST : REF $STR$DESCRIPTOR,
914 2426 1                                     ! Final destination for input chars
915 2427 1                                     FLAGS, ! Enhancement flag
916 2428 1                                     PUT_HERE : REF BLOCK [8, BYTE],
917 2429 1                                     ! Contains input characters
918 2430 1                                     CHARS_READ ! # of input characters
919 2431 1                                     ) =
920 2432 1
921 2433 1 ++
922 2434 1 FUNCTIONAL DESCRIPTION:
923 2435 1
924 2436 1 Convert TEXT input string to Floating or Double Floating Point.
925 2437 1 Do nothing about errors in this routine, return control to calling
926 2438 1 routine.
927 2439 1
928 2440 1 FORMAL PARAMETERS:
929 2441 1
930 2442 1 STRING_DEST.mt.ds Address of descriptor to receive the read input.
931 2443 1
932 2444 1 FLAGS.rlu.v Screen enhancement flag;
933 2445 1
934 2446 1 PUT_HERE.rt.dx Buffer to hold input characters.
935 2447 1
936 2448 1 CHARS_READ.rlu.v Number of characters accepted as input.
937 2449 1
938 2450 1 IMPLICIT INPUTS:
939 2451 1
940 2452 1 NONE
941 2453 1
942 2454 1 IMPLICIT OUTPUTS:
943 2455 1
944 2456 1 NONE
945 2457 1
946 2458 1 ROUTINE VALUE:
947 2459 1
948 2460 1 1 = Conversion Success
949 2461 1 0 = Conversion Failure
950 2462 1
951 2463 1 SIDE EFFECTS:
952 2464 1
953 2465 1 Signals COB$_INVARG if the syntax of the number is wrong.
954 2466 1 --
955 2467 1
956 2468 2 BEGIN
957 2469 2 LOCAL
958 2470 2 CONV_OK : INITIAL (0) ; ! Conversion flag
959 2471 2 ! =0 error, =1 no error
960 2472 2 LABEL
961 2473 2 FLOAT_PROCESSOR ;
962 2474 2
963 2475 2 FLOAT_PROCESSOR:
964 2476 2 BEGIN
965 2477 2 LOCAL
966 2478 2 TEMP_PUT_HERE : BLOCK [12,BYTE] VOLATILE ;
967 2479 2 ! Temporary buffer with length
968 2480 2 ! reflecting number of chars read
```

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$FLOAT\_CONV - Convert to Floating Point

M 14  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 27  
(5)

969 2481 3  
970 2482 3  
971 2483 3  
972 2484 3  
973 2485 3  
974 2486 3  
975 2487 3  
976 2488 3  
977 2489 3  
978 2490 3  
979 2491 3  
980 2492 3  
981 2493 3  
982 2494 4  
983 2495 4  
984 2496 4  
985 2497 4  
986 2498 4  
987 2499 4  
988 2500 3  
989 2501 3  
990 2502 3  
991 2503 3  
992 2504 3  
993 2505 3  
994 2506 3  
995 2507 3  
996 2508 3  
997 2509 3  
998 2510 3  
999 2511 3  
1000 2512 3  
1001 2513 3  
1002 2514 3  
1003 2515 3  
1004 2516 3  
1005 2517 3  
1006 2518 3  
1007 2519 4  
1008 2520 4  
1009 2521 4  
1010 2522 5  
1011 2523 5  
1012 2524 5  
1013 2525 5  
1014 2526 5  
1015 2527 6  
1016 2528 6  
1017 2529 6  
1018 2530 6  
1019 2531 6  
1020 2532 5  
1021 2533 5  
1022 2534 5  
1023 2535 5  
1024 2536 5  
1025 2537 5

```
LITERAL
  ONLY_E_ALLOWED = 15 ;      ! Bit 0 - blanks are ignored
                              ! Bit 1 - only E or e for exp
                              ! Bit 2 - underflow is an error
                              ! Bit 3 - do not round

BIND
  COMMA = UPLIT (','),
  DOT = UPLIT ('.');

!+
! Pick appropriate conversion routine based on
! data type.
!-
BIND ROUTINE CVTTX = (
  IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_F
  THEN OT$SCVT_T_F

  ELSE IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_D
  THEN OT$SCVT_T_D
  ELSE 0 );

!+
! Make TEMP_PUT_HERE a fixed length descriptor -
! don't need STR$GET1_DX or STR$FREE1_DX.
!-
TEMP_PUT_HERE [DSC$W_LENGTH] = .CHARS_READ ;
TEMP_PUT_HERE [DSC$B_DTYPE] = DSC$K_DTYPE_NL ;
TEMP_PUT_HERE [DSC$B_CLASS] = DSC$K_CLASS_S ;
TEMP_PUT_HERE [DSC$A_POINTER] = .PUT_HERE [DSC$A_POINTER] ;

!+
! If DECIMAL POINT IS COMMA is set and a comma came in,
! must change it to a decimal point before the convert.
! OT$ routines expect a decimal point.
! Also must not allow a dec pt as input in that case.
!-
IF (.FLAGS AND V_DEC_PT) NEQ 0
THEN
  BEGIN
    INCR PTR FROM 0 TO (.CHARS_READ - 1) DO
    BEGIN
      IF CH$EQL (1, .TEMP_PUT_HERE [DSC$A_POINTER] + .PTR,
        1, DOT)
      THEN
        BEGIN
          CONV_OK = 0;      ! Illegal, looking
          LEAVE FLOAT_PROCESSOR; ! for a comma
        END;
      IF CH$EQL (1, .TEMP_PUT_HERE [DSC$A_POINTER] + .PTR,
        1, COMMA)
      THEN
        CH$MOVE (1, DOT, .TEMP_PUT_HERE [DSC$A_POINTER] + .PTR);
```

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$FLOAT\_CONV - Convert to Floating Point

N 14  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 28  
(5)

```

1026 2538 4      END;
1027 2539 4      END
1028 2540 4      !+
1029 2541 4      ! If DECIMAL POINT IS COMMA is NOT set and a comma
1030 2542 4      ! came in, this is a conversion error
1031 2543 4      !-
1032 2544 3      ELSE
1033 2545 4      BEGIN
1034 2546 4          INCR PTR FROM 0 TO (.CHARS_READ - 1) DO
1035 2547 4          BEGIN
1036 2548 5              IF CH$EQL (1, .TEMP_PUT_HERE [DSC$A_POINTER] + .PTR,
1037 2549 5                  1, COMMA)
1038 2550 5              THEN
1039 2551 5                  BEGIN
1040 2552 5                      CONV_OK = 0; ! Illegal, looking
1041 2553 6                      LEAVE FLOAT_PROCESSOR; ! for a decimal pt
1042 2554 6
1043 2555 6                      END;
1044 2556 6                  END ;
1045 2557 6              END ;
1046 2558 5          END ;
1047 2559 4          END ;
1048 2560 3      !+
1049 2561 3      ! Check that input data is not out of range.
1050 2562 3      !-
1051 2563 3      BEGIN
1052 2564 3      LOCAL
1053 2565 4          MAX ; ! Maximum significant digits allowed
1054 2566 4
1055 2567 4          MAX = ( IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_F
1056 2568 4              THEN 7 ! Floating
1057 2569 5              ELSE 16 ) ; ! Double Floating
1058 2570 5
1059 2571 4          CONV_OK = COB$$VERIFY_FL_RANGE ( TEMP_PUT_HERE, .CHARS_READ,
1060 2572 4              .MAX ) ;
1061 2573 4          IF .CONV_OK
1062 2574 4          THEN
1063 2575 4              BEGIN
1064 2576 4                  !+
1065 2577 5                  ! Convert to float or double.
1066 2578 5                  !-
1067 2579 5                  IF NOT ( CVTTX ( TEMP_PUT_HERE,
1068 2580 5                      .STRING_DEST[DSC$A_POINTER],
1069 2581 6                      0, 0, ONLY_E_ALLOWED ) )
1070 2582 6                  THEN CONV_OK = 0 ! Conversion error
1071 2583 6                  ELSE
1072 2584 5                      CONV_OK = 1 ; ! No error
1073 2585 5                  END ;
1074 2586 5              END ;
1075 2587 4          END ;
1076 2588 3      END ;
1077 2589 2      END ;
1078 2590 2      RETURN .CONV_OK ;
1079 2591 2      END ;
1080 2592 1      ! End of COB$$FLOAT_CONV
```

```
00 00 00 2C 004E5 .BLKB 3
00 00 00 2E 004E8 P.AAB: .ASCII \,\<0><0><0>
00 00 00 2E 004EC P.AAC: .ASCII \,\<0><0><0>

COMMA= P.AAB
DOT= P.AAC
```

```
007C 00000 COB$$FLOAT_CONV:
      .WORD Save R2,R3,R4,R5,R6
56      F7 AF 9E 00002 MOVAB DOT, R6
5E      0C C2 00006 SUBL2 #12, SP
      53 D4 00009 CLRL CONV_OK
52      04 AC D0 0000B MOVL STRING_DEST, R2
      54 D4 0000F CLRL R4
0A      02 A2 91 00011 CMPB 2(R2), #10
      0B 12 00015 BNEQ 1$
      54 D6 00017 INCL R4
50 00000000G 00 9E 00019 MOVAB OTSSCVT_T_F, R0
      11 11 00020 BRB 3$
0B      02 A2 91 00022 1$: CMPB 2(R2), #11
      09 12 00026 BNEQ 2$
50 00000000G 00 9E 00028 MOVAB OTSSCVT_T_D, R0
      02 11 0002F BRB 3$
      50 D4 00031 2$: CLRL R0
      50 D0 00033 3$: MOVL R0, R5
02      6E 10 AC B0 00036 MOVW CHARS_READ, TEMP_PUT_HERE
03      AE 10 90 0003A MOVB #16, TEMP_PUT_HERE+2
      50 0C AC D0 00042 MOVB #1, TEMP_PUT_HERE+3
04      AE 04 A0 D0 00046 MOVL PUT_HERE, R0
10      AC 01 C3 0004B MOVL 4(R0), TEMP_PUT_HERE+4
1F      08 AC 06 E1 00050 SUBL3 #1, CHARS_READ, R1
      50 01 CE 00055 BBC #6, FLAGS, 6$
      14 11 00058 MNEGL #1, PTR
      66 04 BE40 91 0005A 4$: BRB 5$
      52 13 0005F CMPB @TEMP_PUT_HERE+4[PTR], DOT
FC      A6 04 BE40 91 00061 BEQL 12$
      05 12 00067 CMPB @TEMP_PUT_HERE+4[PTR], COMMA
04 BE40 66 90 00069 BNEQ 5$
E8      50 51 F3 0006E 5$: MOVB DOT, @TEMP_PUT_HERE+4[PTR]
      11 11 00072 AOBLEQ R1, PTR, 4$
      50 01 CE 00074 6$: BRB 9$
      08 11 00077 MNEGL #1, PTR
FC      A6 04 BE40 91 00079 7$: BRB 8$
      32 13 0007F CMPB @TEMP_PUT_HERE+4[PTR], COMMA
F4      50 51 F3 00081 8$: BEQL 12$
      05 54 E9 00085 9$: AOBLEQ R1, PTR, 7$
      50 07 D0 00088 BLBC R4, 10$
      03 11 0008B MOVL #7, MAX
      50 10 D0 0008D 10$: BRB 11$
      10 50 DD 00090 11$: MOVL #16, MAX
      08 AC DD 00092 PUSHL MAX
0000V CF 03 9F 00095 PUSHL CHARS_READ
      53 03 FB 00098 PUSHAB TEMP_PUT_HERE
      50 D0 0009D CALLS #3, COB$$VERIFY_FL_RANGE
      MOV R0, CONV_OK
```

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$FLOAT\_CONV - Convert to Floating Point

C 15  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 30  
(5)

17	53	E9	000A0	BLBC	CONV_OK, 14\$	: 2575
	0F	DD	000A3	PUSHL	#15	: 2581
	7E	7C	000A5	CLRQ	-(SP)	: 2582
04	A2	D0	000A7	PUSHL	4(R2)	: 2581
10	AE	9F	000AA	PUSHAB	TEMP_PUT_HERE	: 2584
65	05	FB	000AD	CALLS	#5, (R5)-	: 2586
04	50	E8	000B0	BLBS	R0, 13\$	: 2591
	53	D4	000B3	CLRL	CONV_OK	: 2592
	03	11	000B5	BRB	14\$	
53	01	D0	000B7	MOVL	#1, CONV_OK	
50	53	D0	000BA	MOVL	CONV_OK, -R0	
	04	000BD	RET			

; Routine Size: 190 bytes, Routine Base: \_COB\$CODE + 04F0



```
1082 2593 1 %SBTTL 'COB$$$COMP_CONV - Convert to COMP and COMP3'
1083 2594 1 ROUTINE COB$$$COMP_CONV (STRING_DEST : REF $STR$DESCRIPTOR,
1084 2595 1                                     | Final destination for input chars
1085 2596 1                                     | Enhancement flag
1086 2597 1                                     | REF $STR$DESCRIPTOR,
1087 2598 1                                     | REF BLOCK [8, BYTE],
1088 2599 1                                     | Contains input characters
1089 2600 1                                     | # of input characters
1090 2601 1                                     | =1 if DEFAULT value is used
1091 2602 1                                     | =1 if sign should be included
1092 2603 1                                     |
1093 2604 1                                     |
1094 2605 1                                     |
1095 2606 1                                     |
1096 2607 1                                     |
1097 2608 1                                     |
1098 2609 1                                     |
1099 2610 1                                     |
1100 2611 1                                     |
1101 2612 1                                     |
1102 2613 1                                     |
1103 2614 1                                     |
1104 2615 1                                     |
1105 2616 1                                     |
1106 2617 1                                     |
1107 2618 1                                     |
1108 2619 1                                     |
1109 2620 1                                     |
1110 2621 1                                     |
1111 2622 1                                     |
1112 2623 1                                     |
1113 2624 1                                     |
1114 2625 1                                     |
1115 2626 1                                     |
1116 2627 1                                     |
1117 2628 1                                     |
1118 2629 1                                     |
1119 2630 1                                     |
1120 2631 1                                     |
1121 2632 1                                     |
1122 2633 1                                     |
1123 2634 1                                     |
1124 2635 1                                     |
1125 2636 1                                     |
1126 2637 1                                     |
1127 2638 1                                     |
1128 2639 1                                     |
1129 2640 1                                     |
1130 2641 1                                     |
1131 2642 1                                     |
1132 2643 1                                     |
1133 2644 1                                     |
1134 2645 1                                     |
1135 2646 1                                     |
1136 2647 1                                     |
1137 2648 1                                     |
1138 2649 2 BEGIN
```

FUNCTIONAL DESCRIPTION:

Convert TEXT input string to appropriate VAX COBOL COMP or COMP3 data type.  
Do nothing about errors in this routine, return control to calling routine.

FORMAL PARAMETERS:

STRING\_DEST.mt.ds Address of descriptor to receive the read input.  
FLAGS.rlu.v Screen enhancement flag;  
DEFAULT.rt.dx Default source moved to destination descriptor (STRING\_DEST) in the event of null input.  
PUT\_HERE.rt.dx Buffer to hold input characters.  
CHARS\_READ.rlu.v Number of characters accepted as input.  
YES\_DEFAULT.rlu.v Flag = 1 if DEFAULT used because of null input.  
YES\_SIGN.rlu.v Flag = 1 if sign should be included in COMP or COMP3 data type.

IMPLICIT INPUTS:

NONE

IMPLICIT OUTPUTS:

NONE

ROUTINE VALUE:

1 = Conversion Success  
0 = Conversion Failure

SIDE EFFECTS:

-- Signals COB\$\_INVARG if the syntax of the number is wrong.

```

1139 2650 2 LOCAL
1140 2651 2 CONV_OK : INITIAL (0) ;          ! Conversion flag,
1141 2652 2                                     ! =0 error, =1 no error
1142 2653 2 BEGIN
1143 2654 2
1144 2655 2 LOCAL
1145 2656 2     EXPONENT: INITIAL (0),      ! Exponent for the CIT
1146 2657 2     I VALUE: VECTOR[12, BYTE], ! COBOL intermediate temporary
1147 2658 2     SIGN: BYTE,                  ! Sign of the input string
1148 2659 2     SEND_CHARS_READ;              ! Local to hold .CHARS_READ
1149 2660 2
1150 2661 2
1151 2662 2     + COB$$$STRIP_BLANKS_SIGN may change the value of
1152 2663 2     SEND_CHARS_READ therefore it is important to send
1153 2664 2     this local instead of CHARS_READ (which should NEVER
1154 2665 2     be altered).
1155 2666 2
1156 2667 2     SEND_CHARS_READ = .CHARS_READ ;
1157 2668 2
1158 2669 2     + First must strip off leading and trailing blanks and
1159 2670 2     the sign because COB$CVTTI won't accept them.
1160 2671 2
1161 2672 2     IF .YES_DEFAULT
1162 2673 2     THEN
1163 2674 2
1164 2675 2         + This move must be done because STRIP routine
1165 2676 2         writes back into the 1st parameter and DEFAULT
1166 2677 2         is read-only.
1167 2678 2
1168 2679 2         CH$MOVE (.CHARS_READ, .DEFAULT [DSC$A_POINTER],
1169 2680 2             .PUT_HERE [DSC$A_POINTER]);
1170 2681 2
1171 2682 2     CONV_OK = COB$$$STRIP_BLANKS_SIGN (.PUT_HERE, .STRING_DEST,
1172 2683 2         SEND_CHARS_READ, EXPONENT, SIGN, .FLAGS);
1173 2684 2
1174 2685 2     IF .CONV_OK
1175 2686 2     THEN
1176 2687 2
1177 2688 2         + Convert the stripped input string to CIT.
1178 2689 2         Must do convert to CIT, THEN to destination data
1179 2690 2         type because these COBOL conversion routines take
1180 2691 2         into account the scale factor.
1181 2692 2
1182 2693 2         CONV_OK = COB$CVTTI R8 (.SEND_CHARS_READ,
1183 2694 2             .PUT_HERE [DSC$A_POINTER], I_VALUE);
1184 2695 2
1185 2696 2     IF .CONV_OK
1186 2697 2     THEN
1187 2698 2         BEGIN
1188 2699 2
1189 2700 2             LOCAL
1190 2701 2             SCALE;
1191 2702 2
1192 2703 2             + Pick the appropriate conversion routine based
1193 2704 2             on data type. Note that routines have the
1194 2705 2             same linkage. At this point we can be sure that
1195 2706 2
```

1196	2707	4
1197	2708	4
1198	2709	5
1199	2710	5
1200	2711	5
1201	2712	5
1202	2713	5
1203	2714	5
1204	2715	5
1205	2716	5
1206	2717	5
1207	2718	5
1208	2719	5
1209	2720	5
1210	2721	5
1211	2722	5
1212	2723	5
1213	2724	5
1214	2725	5
1215	2726	5
1216	2727	4
1217	2728	4
1218	2729	4
1219	2730	4
1220	2731	4
1221	2732	4
1222	2733	4
1223	2734	4
1224	2735	4
1225	2736	4
1226	2737	4
1227	2738	4
1228	2739	4
1229	2740	4
1230	2741	4
1231	2742	4
1232	2743	4
1233	2744	4
1234	2745	4
1235	2746	4
1236	2747	4
1237	2748	4
1238	2749	4
1239	2750	4
1240	2751	4
1241	2752	4
1242	2753	5
1243	2754	5
1244	2755	4
1245	2756	4
1246	2757	5
1247	2758	5
1248	2759	5
1249	2760	5
1250	2761	5
1251	2762	5
1252	2763	5

we are dealing with one of these six data types.

```

BIND ROUTINE CVTIX = (
  IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_W
  THEN COB$CVTIW_R8

  ELSE IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_WU
  THEN COB$CVTIW_R8

  ELSE IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_L
  THEN COB$CVTIL_R8

  ELSE IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_LU
  THEN COB$CVTIL_R8

  ELSE IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_Q
  THEN COB$CVTIQ_R8

  ELSE IF .STRING_DEST [DSC$B_DTYPE] EQL DSC$K_DTYPE_QU
  THEN COB$CVTIQ_R8
  ELSE 0) : JSB_878 ;
```

+ First must re-insert the sign in the CIT.  
The sign of the CIT is contained in byte 12  
(see appendix C of the RTL Ref Manual for a  
description of the CIT). 'C' means + and 'D'  
means -. COB\$CVTII always returns a positive  
number, so if the input number was really  
negative, must make the sign byte negative.  
Check to see if sign should be included.

```

IF .SIGN EQL %C'-' AND .YES_SIGN
THEN
  I_VALUE[11] = .I_VALUE[11] + 1;
```

+ Next must insert the exponent in the CIT.  
The first word of the CIT contains the exponent.

```

CH$MOVE (2, EXPONENT, I_VALUE);
```

+ Convert from CIT to destination data type taking  
into account the scale factor.

```

SCALE = ( IF .STRING_DEST [DSC$B_CLASS] EQL DSC$K_CLASS_SD
  THEN -.STRING_DEST [DSC$B_SCALE]
  ELSE 0 );
```

```

CONV_OK = ( IF CVTIX EQL 0
  THEN
```

+ Packed

```

  COB$CVTIP R9 (.SCALE, I_VALUE,
    .STRING_DEST [DSC$W_LENGTH], .STRING_DEST [DSC$A_POINTER])
```

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$COMP\_CONV - Convert to COMP and COMP3

G 15  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 34  
(6)

: 1253  
: 1254  
: 1255  
: 1256  
: 1257  
: 1258  
: 1259  
: 1260  
: 1261  
: 1262

2764 5  
2765 5  
2766 5  
2767 5  
2768 4  
2769 3  
2770 2  
2771 2  
2772 2  
2773 1

RETURN .CONV\_OK ;  
END ;

END;  
END ;

ELSE

Integer types

CVTIX (.SCALE, I\_VALUE, .STRING\_DEST [DSC\$A\_POINTER]) ) ;

! End of COB\$\$COMP\_CONV

OFFC 00000 COB\$\$COMP CONV:

	SE	18	C2	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	: 2594
		5B	D4	00005	SUBL2	#24, SP	
		AE	D4	00007	CLRL	CONV_OK	: 2649
	08	14	AC	D0	0000A	CLRL	: 2653
		18	AC	E9	0000F	EXPOENT	: 2667
	OF	18	AC	D0	00013	CHARS_READ, SEND_CHARS_READ	: 2673
	51	0C	AC	D0	00017	YES_DEFAULT, 1\$	: 2680
	50	10	AC	D0	00017	DEFAULT, R1	: 2681
04	B0	14	AC	D0	0001B	PUT_HERE, R0	
	04	08	AC	DD	00022	CHARS_READ, @4(R1), @4(R0)	
		04	AE	9F	00025	1\$: PUSHL	: 2684
		0C	AE	9F	00028	FLAGS	: 2683
		14	AE	9F	0002B	SIGN	
	SA	04	AC	D0	0002E	EXPONENT	
		SA	DD	00032	PUSHAB	SEND_CHARS_READ	
	52	10	AC	D0	00034	STRING_DEST, R10	
		52	DD	00038	PUSHL	R10	
0000V	CF	06	FB	0003A	MOV	PUT_HERE, R2	
	5B	50	D0	0003F	PUSHL	R2	
	15	5B	E9	00042	CALLS	#6, COB\$\$STRIP_BLANKS_SIGN	
	58	0C	AE	9E	00045	MOV	: 2685
	57	04	A2	D0	00049	R0, CONV_OK	: 2693
	56	08	AE	D0	0004D	CONV_OK, -2\$	
		00	16	00051	MOVAB	I_VALUE, R8	
	5B	50	D0	00057	MOV	4(R2), R7	
	03	5B	E8	0005A	2\$: MOV	SEND_CHARS_READ, R6	
		009D	31	0005D	JSB	COB\$CVTIW_R8	
	50	02	AA	9A	00060	3\$: MOV	: 2696
	07	50	91	00064	BRB	CONV_OK, -3\$	: 2710
		09	12	00067	BRW	18\$	
	51	00000000G	00	9E	00069	MOVZBL	: 2713
		45	11	00070	CMPB	R0, #7	
	03	50	91	00072	BNEQ	4\$	
		09	12	00075	MOVAB	COB\$CVTIW_R8, R1	
	51	00000000G	00	9E	00077	BRB	: 2716
		1F	11	0007E	BRB	8\$	
	08	50	91	00080	5\$: CMPB	R0, #8	
		09	12	00083	BNEQ	7\$	
	51	00000000G	00	9E	00085	6\$: MOVAB	: 2719
		11	11	0008C	BRB	8\$	
	04	50	91	0008E	7\$: CMPB	R0, #4	

COBSACCECV  
1-001

COBSACCECV - ACCEPT Conversion routines  
COB\$\$COMP\_CONV - Convert to COMP and COMP3

H 15  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 35  
(6)

		F2	13	00091	BEQL	6\$		
09		50	91	00093	CMPB	R0, #9		2722
		0C	12	00096	BNEQ	9\$		
51	00000000G	00	9E	00098	MOVAB	COB\$CVTIQ_R8, R1		
50		51	D0	0009F	8\$:	MOV L	R1, R0	
		10	11	000A2	BRB	11\$		
05		50	91	000A4	9\$:	CMPB	R0, #5	2725
		09	12	000A7	BNEQ	10\$		
50	00000000G	00	9E	000A9	MOVAB	COB\$CVTIQ_R8, R0		
		02	11	000B0	BRB	11\$		
		50	D4	000B2	10\$:	CLRL	R0	
51		50	D0	000B4	11\$:	MOV L	R0, R1	2713
2D		6E	91	000B7	12\$:	CMPB	SIGN, #45	2739
		07	12	000BA	BNEQ	13\$		
03	1C	AC	E9	000BC	BLBC	YES SIGN, 13\$		
	17	AE	96	000C0	INCB	I VALUE+11		2741
0C	AE	04	AE	80	000C3	13\$:	MOVW	EXPONENT, I_VALUE
	09	03	AA	91	000C8	CMPB	3(R10), #9	2747
		09	12	000CC	BNEQ	14\$		2753
56	08	AA	98	000CE	CVTBL	8(R10), SCALE		2754
56		56	CE	000D2	MNEGL	SCALE, SCALE		
		02	11	000D5	BRB	15\$		
		56	D4	000D7	14\$:	CLRL	SCALE	2753
		51	D5	000D9	15\$:	TSTL	R1	2757
		13	12	000DB	BNEQ	16\$		
57	0C	AE	9E	000DD	MOVAB	I VALUE, R7		2762
59	04	AA	D0	000E1	MOVL	4(R10), R9		
58		6A	3C	000E5	MOVZWL	(R10), R8		
	00000000G	00	16	000E8	JSB	COB\$CVTIP_R9		
		0A	11	000EE	BRB	17\$		
57	0C	AE	9E	000F0	16\$:	MOVAB	I VALUE, R7	2768
58	04	AA	D0	000F4	MOVL	4(R10), R8		
		61	16	000F8	JSB	(R1)		
58		50	D0	000FA	17\$:	MOVL	R0, CONV_OK	
50		5B	D0	000FD	18\$:	MOVL	CONV_OK, R0	2772
		04	00100		RET			2773

; Routine Size: 257 bytes, Routine Base: \_COB\$CODE + 05AE

```
1264 2774 1 %SBTTL 'COB$$$STRIP_BLANKS_SIGN - Pull blanks and sign'
1265 2775 1 ROUTINE COB$$$STRIP_BLANKS_SIGN (
1266 2776 1
1267 2777 1     INPUT_STRING      : REF $STR$DESCRIPTOR,
1268 2778 1     STRING_DEST     : REF $STR$DESCRIPTOR,
1269 2779 1     NUM_DIGITS,
1270 2780 1
1271 2781 1     EXPONENT,
1272 2782 1     SIGN_VAL         : REF BLOCK [ , BYTE],
1273 2783 1     FLAGS           ) =
1274 2784 1
1275 2785 1
1276 2786 1
1277 2787 1 ++
1278 2788 1 FUNCTIONAL DESCRIPTION:
1279 2789 1
1280 2790 1     Strips leading and trailing blanks from the input numeric string.
1281 2791 1     Puts the remaining string back and adjusts the number of digits
1282 2792 1     accordingly. Also strips off the sign char and puts it into the
1283 2793 1     output parameter SIGN_VAL.
1284 2794 1     Strips the decimal point and figures out the exponent.
1285 2795 1     Does nothing about errors in this routine, returns control to
1286 2796 1     calling routine.
1287 2797 1
1288 2798 1 FORMAL PARAMETERS:
1289 2799 1
1290 2800 1     INPUT_STRING.rt.dx  The numeric string to scan
1291 2801 1     STRING_DESC.rt.dx  The destination string
1292 2802 1     NUM_DIGITS.ml.r    As input, its the number of chars read
1293 2803 1                     As output, its the number of chars after
1294 2804 1                     the stripping
1295 2805 1     EXPONENT.wl.r      The exponent
1296 2806 1     SIGN_VAL.wb.r      The sign character
1297 2807 1
1298 2808 1 IMPLICIT INPUTS:
1299 2809 1
1300 2810 1     NONE
1301 2811 1
1302 2812 1 IMPLICIT OUTPUTS:
1303 2813 1
1304 2814 1     NONE
1305 2815 1
1306 2816 1 ROUTINE VALUE:
1307 2817 1
1308 2818 1     0 = failure, 1 = success
1309 2819 1
1310 2820 1 SIDE EFFECTS:
1311 2821 1
1312 2822 1     NONE
1313 2823 1
1314 2824 2 BEGIN
1315 2825 2
1316 2826 2 LOCAL
1317 2827 2     TEMP_NUM_DIGITS : INITIAL (0),
1318 2828 2     BUF_DESC       : BLOCK [R, BYTE] VOLATILE,
1319 2829 2     SIGN_SEEN      : INITIAL (0),
1320 2830 2     DIGIT_SEEN     : INITIAL (0),
1321 2831 2
1322 2832 2
1323 2833 2
1324 2834 2
1325 2835 2
1326 2836 2
1327 2837 2
1328 2838 2
1329 2839 2
1330 2840 2
1331 2841 2
1332 2842 2
1333 2843 2
1334 2844 2
1335 2845 2
1336 2846 2
1337 2847 2
1338 2848 2
1339 2849 2
1340 2850 2
1341 2851 2
1342 2852 2
1343 2853 2
1344 2854 2
1345 2855 2
1346 2856 2
1347 2857 2
1348 2858 2
1349 2859 2
1350 2860 2
1351 2861 2
1352 2862 2
1353 2863 2
1354 2864 2
1355 2865 2
1356 2866 2
1357 2867 2
1358 2868 2
1359 2869 2
1360 2870 2
1361 2871 2
1362 2872 2
1363 2873 2
1364 2874 2
1365 2875 2
1366 2876 2
1367 2877 2
1368 2878 2
1369 2879 2
1370 2880 2
1371 2881 2
1372 2882 2
1373 2883 2
1374 2884 2
1375 2885 2
1376 2886 2
1377 2887 2
1378 2888 2
1379 2889 2
1380 2890 2
1381 2891 2
1382 2892 2
1383 2893 2
1384 2894 2
1385 2895 2
1386 2896 2
1387 2897 2
1388 2898 2
1389 2899 2
1390 2900 2
1391 2901 2
1392 2902 2
1393 2903 2
1394 2904 2
1395 2905 2
1396 2906 2
1397 2907 2
1398 2908 2
1399 2909 2
1400 2910 2
1401 2911 2
1402 2912 2
1403 2913 2
1404 2914 2
1405 2915 2
1406 2916 2
1407 2917 2
1408 2918 2
1409 2919 2
1410 2920 2
1411 2921 2
1412 2922 2
1413 2923 2
1414 2924 2
1415 2925 2
1416 2926 2
1417 2927 2
1418 2928 2
1419 2929 2
1420 2930 2
1421 2931 2
1422 2932 2
1423 2933 2
1424 2934 2
1425 2935 2
1426 2936 2
1427 2937 2
1428 2938 2
1429 2939 2
1430 2940 2
1431 2941 2
1432 2942 2
1433 2943 2
1434 2944 2
1435 2945 2
1436 2946 2
1437 2947 2
1438 2948 2
1439 2949 2
1440 2950 2
1441 2951 2
1442 2952 2
1443 2953 2
1444 2954 2
1445 2955 2
1446 2956 2
1447 2957 2
1448 2958 2
1449 2959 2
1450 2960 2
1451 2961 2
1452 2962 2
1453 2963 2
1454 2964 2
1455 2965 2
1456 2966 2
1457 2967 2
1458 2968 2
1459 2969 2
1460 2970 2
1461 2971 2
1462 2972 2
1463 2973 2
1464 2974 2
1465 2975 2
1466 2976 2
1467 2977 2
1468 2978 2
1469 2979 2
1470 2980 2
1471 2981 2
1472 2982 2
1473 2983 2
1474 2984 2
1475 2985 2
1476 2986 2
1477 2987 2
1478 2988 2
1479 2989 2
1480 2990 2
1481 2991 2
1482 2992 2
1483 2993 2
1484 2994 2
1485 2995 2
1486 2996 2
1487 2997 2
1488 2998 2
1489 2999 2
1490 3000 2
1491 3001 2
1492 3002 2
1493 3003 2
1494 3004 2
1495 3005 2
1496 3006 2
1497 3007 2
1498 3008 2
1499 3009 2
1500 3010 2
1501 3011 2
1502 3012 2
1503 3013 2
1504 3014 2
1505 3015 2
1506 3016 2
1507 3017 2
1508 3018 2
1509 3019 2
1510 3020 2
1511 3021 2
1512 3022 2
1513 3023 2
1514 3024 2
1515 3025 2
1516 3026 2
1517 3027 2
1518 3028 2
1519 3029 2
1520 3030 2
1521 3031 2
1522 3032 2
1523 3033 2
1524 3034 2
1525 3035 2
1526 3036 2
1527 3037 2
1528 3038 2
1529 3039 2
1530 3040 2
1531 3041 2
1532 3042 2
1533 3043 2
1534 3044 2
1535 3045 2
1536 3046 2
1537 3047 2
1538 3048 2
1539 3049 2
1540 3050 2
1541 3051 2
1542 3052 2
1543 3053 2
1544 3054 2
1545 3055 2
1546 3056 2
1547 3057 2
1548 3058 2
1549 3059 2
1550 3060 2
1551 3061 2
1552 3062 2
1553 3063 2
1554 3064 2
1555 3065 2
1556 3066 2
1557 3067 2
1558 3068 2
1559 3069 2
1560 3070 2
1561 3071 2
1562 3072 2
1563 3073 2
1564 3074 2
1565 3075 2
1566 3076 2
1567 3077 2
1568 3078 2
1569 3079 2
1570 3080 2
1571 3081 2
1572 3082 2
1573 3083 2
1574 3084 2
1575 3085 2
1576 3086 2
1577 3087 2
1578 3088 2
1579 3089 2
1580 3090 2
1581 3091 2
1582 3092 2
1583 3093 2
1584 3094 2
1585 3095 2
1586 3096 2
1587 3097 2
1588 3098 2
1589 3099 2
1590 3100 2
1591 3101 2
1592 3102 2
1593 3103 2
1594 3104 2
1595 3105 2
1596 3106 2
1597 3107 2
1598 3108 2
1599 3109 2
1600 3110 2
1601 3111 2
1602 3112 2
1603 3113 2
1604 3114 2
1605 3115 2
1606 3116 2
1607 3117 2
1608 3118 2
1609 3119 2
1610 3120 2
1611 3121 2
1612 3122 2
1613 3123 2
1614 3124 2
1615 3125 2
1616 3126 2
1617 3127 2
1618 3128 2
1619 3129 2
1620 3130 2
1621 3131 2
1622 3132 2
1623 3133 2
1624 3134 2
1625 3135 2
1626 3136 2
1627 3137 2
1628 3138 2
1629 3139 2
1630 3140 2
1631 3141 2
1632 3142 2
1633 3143 2
1634 3144 2
1635 3145 2
1636 3146 2
1637 3147 2
1638 3148 2
1639 3149 2
1640 3150 2
1641 3151 2
1642 3152 2
1643 3153 2
1644 3154 2
1645 3155 2
1646 3156 2
1647 3157 2
1648 3158 2
1649 3159 2
1650 3160 2
1651 3161 2
1652 3162 2
1653 3163 2
1654 3164 2
1655 3165 2
1656 3166 2
1657 3167 2
1658 3168 2
1659 3169 2
1660 3170 2
1661 3171 2
1662 3172 2
1663 3173 2
1664 3174 2
1665 3175 2
1666 3176 2
1667 3177 2
1668 3178 2
1669 3179 2
1670 3180 2
1671 3181 2
1672 3182 2
1673 3183 2
1674 3184 2
1675 3185 2
1676 3186 2
1677 3187 2
1678 3188 2
1679 3189 2
1680 3190 2
1681 3191 2
1682 3192 2
1683 3193 2
1684 3194 2
1685 3195 2
1686 3196 2
1687 3197 2
1688 3198 2
1689 3199 2
1690 3200 2
1691 3201 2
1692 3202 2
1693 3203 2
1694 3204 2
1695 3205 2
1696 3206 2
1697 3207 2
1698 3208 2
1699 3209 2
1700 3210 2
1701 3211 2
1702 3212 2
1703 3213 2
1704 3214 2
1705 3215 2
1706 3216 2
1707 3217 2
1708 3218 2
1709 3219 2
1710 3220 2
1711 3221 2
1712 3222 2
1713 3223 2
1714 3224 2
1715 3225 2
1716 3226 2
1717 3227 2
1718 3228 2
1719 3229 2
1720 3230 2
1721 3231 2
1722 3232 2
1723 3233 2
1724 3234 2
1725 3235 2
1726 3236 2
1727 3237 2
1728 3238 2
1729 3239 2
1730 3240 2
1731 3241 2
1732 3242 2
1733 3243 2
1734 3244 2
1735 3245 2
1736 3246 2
1737 3247 2
1738 3248 2
1739 3249 2
1740 3250 2
1741 3251 2
1742 3252 2
1743 3253 2
1744 3254 2
1745 3255 2
1746 3256 2
1747 3257 2
1748 3258 2
1749 3259 2
1750 3260 2
1751 3261 2
1752 3262 2
1753 3263 2
1754 3264 2
1755 3265 2
1756 3266 2
1757 3267 2
1758 3268 2
1759 3269 2
1760 3270 2
1761 3271 2
1762 3272 2
1763 3273 2
1764 3274 2
1765 3275 2
1766 3276 2
1767 3277 2
1768 3278 2
1769 3279 2
1770 3280 2
1771 3281 2
1772 3282 2
1773 3283 2
1774 3284 2
1775 3285 2
1776 3286 2
1777 3287 2
1778 3288 2
1779 3289 2
1780 3290 2
1781 3291 2
1782 3292 2
1783 3293 2
1784 3294 2
1785 3295 2
1786 3296 2
1787 3297 2
1788 3298 2
1789 3299 2
1790 3300 2
1791 3301 2
1792 3302 2
1793 3303 2
1794 3304 2
1795 3305 2
1796 3306 2
1797 3307 2
1798 3308 2
1799 3309 2
1800 3310 2
1801 3311 2
1802 3312 2
1803 3313 2
1804 3314 2
1805 3315 2
1806 3316 2
1807 3317 2
1808 3318 2
1809 3319 2
1810 3320 2
1811 3321 2
1812 3322 2
1813 3323 2
1814 3324 2
1815 3325 2
1816 3326 2
1817 3327 2
1818 3328 2
1819 3329 2
1820 3330 2
1821 3331 2
1822 3332 2
1823 3333 2
1824 3334 2
1825 3335 2
1826 3336 2
1827 3337 2
1828 3338 2
1829 3339 2
1830 3340 2
1831 3341 2
1832 3342 2
1833 3343 2
1834 3344 2
1835 3345 2
1836 3346 2
1837 3347 2
1838 3348 2
1839 3349 2
1840 3350 2
1841 3351 2
1842 3352 2
1843 3353 2
1844 3354 2
1845 3355 2
1846 3356 2
1847 3357 2
1848 3358 2
1849 3359 2
1850 3360 2
1851 3361 2
1852 3362 2
1853 3363 2
1854 3364 2
1855 3365 2
1856 3366 2
1857 3367 2
1858 3368 2
1859 3369 2
1860 3370 2
1861 3371 2
1862 3372 2
1863 3373 2
1864 3374 2
1865 3375 2
1866 3376 2
1867 3377 2
1868 3378 2
1869 3379 2
1870 3380 2
1871 3381 2
1872 3382 2
1873 3383 2
1874 3384 2
1875 3385 2
1876 3386 2
1877 3387 2
1878 3388 2
1879 3389 2
1880 3390 2
1881 3391 2
1882 3392 2
1883 3393 2
1884 3394 2
1885 3395 2
1886 3396 2
1887 3397 2
1888 3398 2
1889 3399 2
1890 3400 2
1891 3401 2
1892 3402 2
1893 3403 2
1894 3404 2
1895 3405 2
1896 3406 2
1897 3407 2
1898 3408 2
1899 3409 2
1900 3410 2
1901 3411 2
1902 3412 2
1903 3413 2
1904 3414 2
1905 3415 2
1906 3416 2
1907 3417 2
1908 3418 2
1909 3419 2
1910 3420 2
1911 3421 2
1912 3422 2
1913 3423 2
1914 3424 2
1915 3425 2
1916 3426 2
1917 3427 2
1918 3428 2
1919 3429 2
1920 3430 2
1921 3431 2
1922 3432 2
1923 3433 2
1924 3434 2
1925 3435 2
1926 3436 2
1927 3437 2
1928 3438 2
1929 3439 2
1930 3440 2
1931 3441 2
1932 3442 2
1933 3443 2
1934 3444 2
1935 3445 2
1936 3446 2
1937 3447 2
1938 3448 2
1939 3449 2
1940 3450 2
1941 3451 2
1942 3452 2
1943 3453 2
1944 3454 2
1945 3455 2
1946 3456 2
1947 3457 2
1948 3458 2
1949 3459 2
1950 3460 2
1951 3461 2
1952 3462 2
1953 3463 2
1954 3464 2
1955 3465 2
1956 3466 2
1957 3467 2
1958 3468 2
1959 3469 2
1960 3470 2
1961 3471 2
1962 3472 2
1963 3473 2
1964 3474 2
1965 3475 2
1966 3476 2
1967 3477 2
1968 3478 2
1969 3479 2
1970 3480 2
1971 3481 2
1972 3482 2
1973 3483 2
1974 3484 2
1975 3485 2
1976 3486 2
1977 3487 2
1978 3488 2
1979 3489 2
1980 3490 2
1981 3491 2
1982 3492 2
1983 3493 2
1984 3494 2
1985 3495 2
1986 3496 2
1987 3497 2
1988 3498 2
1989 3499 2
1990 3500 2
1991 3501 2
1992 3502 2
1993 3503 2
1994 3504 2
1995 3505 2
1996 3506 2
1997 3507 2
1998 3508 2
1999 3509 2
2000 3510 2
2001 3511 2
2002 3512 2
2003 3513 2
2004 3514 2
2005 3515 2
2006 3516 2
2007 3517 2
2008 3518 2
2009 3519 2
2010 3520 2
2011 3521 2
2012 3522 2
2013 3523 2
2014 3524 2
2015 3525 2
2016 3526 2
2017 3527 2
2018 3528 2
2019 3529 2
2020 3530 2
2021 3531 2
2022 3532 2
2023 3533 2
2024 3534 2
2025 3535 2
2026 3536 2
2027 3537 2
2028 3538 2
2029 3539 2
2030 3540 2
2031 3541 2
2032 3542 2
2033 3543 2
2034 3544 2
2035 3545 2
2036 3546 2
2037 3547 2
2038 3548 2
2039 3549 2
2040 3550 2
2041 3551 2
2042 3552 2
2043 3553 2
2044 3554 2
2045 3555 2
2046 3556 2
2047 3557 2
2048 3558 2
2049 3559 2
2050 3560 2
2051 3561 2
2052 3562 2
2053 3563 2
2054 3564 2
2055 3565 2
2056 3566 2
2057 3567 2
2058 3568 2
2059 3569 2
2060 3570 2
2061 3571 2
2062 3572 2
2063 3573 2
2064 3574 2
2065 3575 2
2066 3576 2
2067 3577 2
2068 3578 2
2069 3579 2
2070 3580 2
2071 3581 2
2072 3582 2
2073 3583 2
2074 3584 2
2075 3585 2
2076 3586 2
2077 3587 2
2078 3588 2
2079 3589 2
2080 3590 2
2081 3591 2
2082 3592 2
2083 3593 2
2084 3594 2
2085 3595 2
2086 3596 2
2087 3597 2
2088 3598 2
2089 3599 2
2090 3600 2
2091 3601 2
2092 3602 2
2093 3603 2
2094 3604 2
2095 3605 2
2096 3606 2
2097 3607 2
2098 3608 2
2099 3609 2
2100 3610 2
2101 3611 2
2102 3612 2
2103 3613 2
2104 3614 2
2105 3615 2
2106 3616 2
2107 3617 2
2108 3618 2
2109 3619 2
2110 3620 2
2111 3621 2
2112 3622 2
2113 3623 2
2114 3624 2
2115 3625 2
2116 3626 2
2117 3627 2
2118 3628 2
2119 3629 2
2120 3630 2
2121 3631 2
2122 3632 2
2123 3633 2
2124 3634 2
2125 3635 2
2126 3636 2
2127 3637 2
2128 3638 2
2129 3639 2
2130 3640 2
2131 3641 2
2132 3642 2
2133 3643 2
2134 3644 2
2135 3645 2
2136 3646 2
2137 3647 2
2138 3648 2
2139 3649 2
2140 3650 2
2141 3651 2
2142 3652 2
2143 3653 2
2144 3654 2
2145 3655 2
2146 3656 2
2147 3657 2
2148 3658 2
2149 3659 2
2150 3660 2
2151 3661 2
2152 3662 2
2153 3663 2
2154 3664 2
2155 3665 2
2156 3666 2
2157 3667 2
2158 3668 2
2159 3669 2
2160 3670 2
2161 3671 2
2162 3672 2
2163 3673 2
2164 3674 2
2165 3675 2
2166 3676 2
2167 3677 2
2168 3678 2
2169 3679 2
2170 3680 2
2171 3681 2
2172 3682 2
2173 3683 2
2174 3684 2
2175 3685 2
2176 3686 2
2177 3687 2
2178 3688 2
2179 3689 2
2180 3690 2
2181 3691 2
2182 3692 2
2183 3693 2
2184 3694 2
2185 3695 2
2186 3696 2
2187 3697 2
2188 3698 2
2189 3699 2
2190 3700 2
2191 3701 2
2192 3702 2
2193 3703 2
2194 3704 2
2195 3705 2
2196 3706 2
2197 3707 2
2198 3708 2
2199 3709 2
2200 3710 2
2201 3711 2
2202 3712 2
2203 3713 2
2204 3714 2
2205 3715 2
2206 3716 2
2207 3717 2
2208 3718 2
2209 3719 2
2210 3720 2
2211 3721 2
2212 3722 2
2213 3723 2
2214 3724 2
2215 3725 2
2216 3726 2
2217 3727 2
2218 3728 2
2219 3729 2
2220 3730 2
2221 3731 2
2222 3732 2
2223 3733 2
2224 3734 2
2225 3735 2
2226 3736 2
2227 3737 2
2228 3738 2
2229 3739 2
2230 3740 2
2231 3741 2
2232 3742 2
2233 3743 2
2234 3744 2
2235 3745 2
2236 3746 2
2237 3747 2
2238 3748 2
2239 3749 2
2240 3750 2
2241 3751 2
2242 3752 2
2243 3753 2
2244 3754 2
224
```

COBSACCECV  
1-001

COBSACCECV - ACCEPT Conversion routines  
COB\$\$\$STRIP\_BLANKS\_SIGN - Pull blanks and sign

J 15  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 37  
7

```

1321 2831 2      DOT SEEN : INITIAL (0),          ! 1 = we have seen a decimal point
1322 2832 2      ZERO SEEN : INITIAL (0),        ! 1 = zero seen
1323 2833 2      BLANKS_SEEN : INITIAL (0),      ! 1 = we have seen trailing blanks
1324 2834 2      PUTTER : INITIAL (0),          ! Counts position in the output buffer
1325 2835 2      LEFT DEC : INITIAL (0),        ! Number of digits to left of dec. pt.
1326 2836 2      PDATA_FLAG: INITIAL (0),      ! Flag to indicate if a P Picture item
1327 2837 2      BUF : REF VECTOR [1100, BYTE], ! Addresses result
1328 2838 2      INP : REF VECTOR [1100, BYTE], ! Addresses result in input_string
1329 2839 2      LEADING ZEROES: INITIAL (0),   ! Counter of leading zeroes
1330 2840 2      ARG : REF VECTOR [1100, BYTE]; ! Addresses source

1331 2841 2
1332 2842 2      LITERAL
1333 2843 2          SUCCESS = 1,
1334 2844 2          FAILURE = 0;
1335 2845 2
1336 2846 2      BIND
1337 2847 2          ZERO = UPLIT ('0');
1338 2848 2
1339 2849 2      + Enable a handler to free the local string in case of an error.
1340 2850 2      -
1341 2851 2
1342 2852 2      ENABLE
1343 2853 2          COB$$$FREE_STRINGS (BUF_DESC);
1344 2854 2
1345 2855 2      +
1346 2856 2      - If there were no digits input, it means that a <CR> was hit.
1347 2857 2
1348 2858 2      IF ..NUM_DIGITS EQL 0
1349 2859 2      THEN
1350 2860 2          RETURN SUCCESS;
1351 2861 2
1352 2862 2      +
1353 2863 2      - Allocate enough space to hold the digits. It is convenient to
1354 2864 2      allocate before scanning, so we may allocate a little too much,
1355 2865 2      but the space will be freed before we return.
1356 2866 2
1357 2867 2      BUF_DESC [DSC$W_LENGTH] = 0;
1358 2868 2      BUF_DESC [DSC$B_DTYPE] = DSC$K_DTYPE_NU;
1359 2869 2      BUF_DESC [DSC$B_CLASS] = DSC$K_CLASS_D;
1360 2870 2      BUF_DESC [DSC$A_POINTER] = 0;
1361 2871 2      STR$GET1_DX (.NUM_DIGITS, BUF_DESC);
1362 2872 2
1363 2873 2      +
1364 2874 2      - Set pointers.
1365 2875 2
1366 2876 2      BUF = .BUF_DESC [DSC$A_POINTER];
1367 2877 2      ARG = .INPOT_STRING [DSC$A_POINTER];
1368 2878 2      .SIGN_VAL = %C'+';
1369 2879 2
1370 2880 2      +
1371 2881 2      - Scan the input number, put result in BUF.
1372 2882 2
1373 2883 2
1374 2884 2      IF NOT ( COB$$$SCAN_INPUT ( .INPUT_STRING, ..NUM_DIGITS, .FLAGS, BUF_DESC,
1375 2885 2          LEFT DEC, TEMP_NUM_DIGITS, .SIGN_VAL, PUTTER,
1376 2886 2          LEADING ZEROES, SIGN_SEEN, DIGIT_SEEN, DOT_SEEN,
1377 2887 2          ZERO_SEEN, BLANKS_SEEN ) )
```

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$STRIP\_BLANKS\_SIGN - Pull blanks and sign

K 15  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 38  
(7)

```

1378 2888 2 THEN
1379 2889 2     RETURN 0 ;
1380 2890 2
1381 2891 2
1382 2892 2
1383 2893 2
1384 2894 2
1385 2895 2
1386 2896 2
1387 2897 2
1388 2898 2
1389 2899 2
1390 2900 2
1391 2901 2
1392 2902 2
1393 2903 2
1394 2904 2
1395 2905 2
1396 2906 2
1397 2907 2
1398 2908 2
1399 2909 2
1400 2910 2
1401 2911 2
1402 2912 2
1403 2913 2
1404 2914 2
1405 2915 2
1406 2916 2
1407 2917 2
1408 2918 2
1409 2919 2
1410 2920 2
1411 2921 2
1412 2922 2
1413 2923 2
1414 2924 2
1415 2925 2
1416 2926 2
1417 2927 2
1418 2928 2
1419 2929 2
1420 2930 2
1421 2931 2
1422 2932 2
1423 2933 2
1424 2934 2
1425 2935 2
1426 2936 2
1427 2937 2
1428 2938 2
1429 2939 2
1430 2940 2
1431 2941 2
1432 2942 2
1433 2943 2
1434 2944 2

      THEN
      RETURN 0 ;

      !+
      !- If there are no digits, or only leading zeros, take the number to
      !- be zero. Don't be too gullible, however.

      IF ( NOT .DIGIT_SEEN )
      THEN
      BEGIN

      IF ( .SIGN_SEEN OR .DOT_SEEN OR .BLANKS_SEEN ) AND ( .ZERO_SEEN EQL 0 )
      THEN RETURN FAILURE ;

      BUF [ .PUTTER ] = %C'0' ;
      PUTTER = .PUTTER + 1 ;
      TEMP_NUM_DIGITS = .TEMP_NUM_DIGITS + 1 ;

      END

      !+
      !- Validate size of entered data, left and right of decimal point.

      ELSE
      BEGIN

      LOCAL
      DEST_LENGTH ,
      OK_LEFT ,
      RIGHT_DEC : INITIAL ( 0 ) ;

      ! Destination length
      ! Correct number of digits allowed
      ! to left of decimal point
      ! Number of digits to right of dec pt

      IF NOT ( .DOT_SEEN )
      THEN LEFT_DEC = .TEMP_NUM_DIGITS ;
      RIGHT_DEC = .TEMP_NUM_DIGITS - .LEFT_DEC ;

      ! No dec pt. therefore
      ! all digits are left_dec

      !+
      !- Strip trailing zeroes after the decimal point.

      INCR GETTER FROM 1 TO .RIGHT_DEC DO
      IF .BUF [ .TEMP_NUM_DIGITS - .GETTER ] EQL %C'0'
      THEN
      RIGHT_DEC = .RIGHT_DEC - 1
      ELSE
      EXITLOOP ;

      DEST_LENGTH = .STRING_DEST [ DSC$W_LENGTH ] ;
      SELECT ONE .STRING_DEST [ DSC$B_CLASS ] OF
      SET
      [ DSC$K_CLASS_S ] :
      BEGIN
      !+

```



COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$STRIP\_BLANKS\_SIGN - Pull blanks and sign

L 15  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 B1 iss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 39  
(7)

```

1435 2945 '      ! If a decimal point was typed in, all the digits after it
1436 2946 4      ! MUST be zeroes.
1437 2947 4      !
1438 2948 4
1439 2949 4      IF .RIGHT_DEC GTR 0
1440 2950 4      THEN
1441 2951 4          INCR I FROM (.PUTTER - .RIGHT_DEC) TO .PUTTER DO
1442 2952 4              IF .BUF[I] NEQ '0'
1443 2953 4                  THEN RETURN FAILURE;
1444 2954 4
1445 2955 4      IF .LEFT_DEC GTR .DEST_LENGTH
1446 2956 4      THEN RETURN FAILURE;          ! Data entered too big
1447 2957 4
1448 2958 4      END ;
1449 2959 3
1450 2960 3      [ DSC$b_CLASS_SD ] :
1451 2961 3      BEGIN
1452 2962 4
1453 2963 4      LOCAL
1454 2964 4          LENGTH_DIFF,          ! Difference between number of digits
1455 2965 4          ! to the left of the decimal point
1456 2966 4          ! in the typed in number and the dest
1457 2967 4          LENGTH_DIFF2;          ! Difference between number of digits
1458 2968 4          ! to the right of the decimal point
1459 2969 4          ! in the typed in number and the dest
1460 2970 4
1461 2971 4      !+
1462 2972 4      ! This is checking for the P Picture of 99PP.
1463 2973 4      ! If the scale is positive and the number of digits in the
1464 2974 4      ! number equal the scale factor, then simply copy the digits
1465 2975 4      ! in BUF to the destination descriptor.
1466 2976 4      ! NOTE: Code for P Picture left in lowercase.
1467 2977 4      !-
1468 2978 4      if .string_dest[dsc$b_scale] gtr 0
1469 2979 4      then
1470 2980 5          begin
1471 2981 5              local
1472 2982 5                  tot_digits,
1473 2983 5                  diff;
1474 2984 5
1475 2985 5              tot_digits = (.string_dest[dsc$b_digits] + .string_dest[dsc$b_scale]);
1476 2986 5              if ?(.right_dec gtr 0) or (.temp_num_digits gtr .tot_digits)
1477 2987 6                  then
1478 2988 5                      ! number too large
1479 2989 5                      return 0;          ! re-prompt - error
1480 2990 5
1481 2991 5              if .temp_num_digits leq .string_dest[dsc$b_scale]
1482 2992 5              then
1483 2993 6                  begin
1484 2994 6                      str$dupl_char (.input_string,temp_num_digits,zero);
1485 2995 6                      pdata_flag = 1;          ! answer is zero
1486 2996 6
1487 2997 6                  end
1488 2998 6              else
1489 2999 5                  begin
1490 3000 6                      !+
1491 3001 6
```

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$STRIP\_BLANKS\_SIGN - Pull blanks and sign

M 15  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 40  
(7)

```

: 1492      3002  6      ! Zero out the destination field using the digits as
: 1493      3003  6      ! proper number of zero fill characters, rather than
: 1494      3004  6      ! using the length as found in the descriptor, since
: 1495      3005  6      ! class SD is a special case.
: 1496      3006  6      !
: 1497      3007  6      str$dupl_char (.input_string,tot_digits,zero);
: 1498      3008  6      if .leading zeroes neq 0
: 1499      3009  6      then
: 1500      3010  7          begin
: 1501      3011  7              diff = (.string_dest[dsc$b_digits] + .string_dest[dsc$b_scale]) - .temp_num_digi
: 1502      3012  7              ch$move (.string_dest[dsc$b_digits],.buf,.input_string[dsc$a_pointer]+.diff);
: 1503      3013  7              end
: 1504      3014  7          else
: 1505      3015  6              begin
: 1506      3016  7                  diff = .temp_num_digits - .string_dest[dsc$b_scale];
: 1507      3017  7                  if .diff eq .string_dest[dsc$b_digits]
: 1508      3018  7                      then
: 1509      3019  7                          ch$move (.diff,.buf,.input_string[dsc$a_pointer])
: 1510      3020  7                      else
: 1511      3021  7                          begin
: 1512      3022  7                              tot_digits = .tot_digits - .temp_num_digits;
: 1513      3023  8                              ch$move (.diff,.buf,.input_string[dsc$a_pointer]+.tot_digits);
: 1514      3024  8                              end;
: 1515      3025  8                          end;
: 1516      3026  8                      end;
: 1517      3027  8                  end;
: 1518      3028  7                  end;
: 1519      3029  7                  pdata_flag = 1;
: 1520      3030  6                  .num_digits = .string_dest[dsc$b_digits] + .string_dest[dsc$b_scale];
: 1521      3031  6                  end;
: 1522      3032  6              end;
: 1523      3033  6          end;
: 1524      3034  5      end
: 1525      3035  5      else
: 1526      3036  5          begin
: 1527      3037  4              if .string_dest[dsc$b_scale] gtr 0
: 1528      3038  5                  then
: 1529      3039  5                      ok_left = .string_dest[dsc$b_digits]
: 1530      3040  5                  else
: 1531      3041  5                      OK_LEFT = .STRING_DEST [DSC$B_DIGITS] + .STRING_DEST [DSC$B_SCALE] ;
: 1532      3042  5                  if .ok_left lss 0
: 1533      3043  5                      then
: 1534      3044  5                          begin
: 1535      3045  5                              ! Here we have a P Picture field of type PP99.
: 1536      3046  5                              ! We know this when OK_LEFT is less than zero.
: 1537      3047  6                              ! It requires some special casing.
: 1538      3048  6                              !
: 1539      3049  6                              local
: 1540      3050  6                                  diff,
: 1541      3051  6                                  ok_right,
: 1542      3052  6                                  buf_ptr;
: 1543      3053  6                                  if .left_dec gtr 0
: 1544      3054  6                                      ! error no '.' entered
: 1545      3055  6
: 1546      3056  6
: 1547      3057  6
: 1548      3058  6
```

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$STRIP\_BLANKS\_SIGN - Pull blanks and sign

N 15

15-Sep-1984 23:49:06

14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 41  
(7)

```
: 1549      3059 6      then                                ! ring bell and reprompt
: 1550      3060 6      return 0;
: 1551      3061 6
: 1552      3062 6
: 1553      3063 6      ok_left = 0;
: 1554      3064 6      ok_right = abs(.string_dest[dsc$b_scale]);
: 1555      3065 6      if .right_dec gtr .ok_right
: 1556      3066 6      then
: 1557      3067 6      return 0;
: 1558      3068 6
: 1559      3069 6      +
: 1560      3070 6      This handles case where the number of digits entered
: 1561      3071 6      is less than the absolute value of the scale factor,
: 1562      3072 6      meaning that the number returned would have to be 0.
: 1563      3073 6      The first part of the if statement takes care of the
: 1564      3074 6      case where the number of digits entered equals the
: 1565      3075 6      number of digits expected taking into account if the
: 1566      3076 6      absolute value of the scale factor is equal to the
: 1567      3077 6      number of digits entered to the right of the decimal
: 1568      3078 6      point thereby giving us a result of zero again.
: 1569      3079 6      -
: 1570      3080 7      diff = (abs(.string_dest[dsc$b_scale]) - .string_dest[dsc$b_digits]);
: 1571      3081 6      if ((.right_dec eql .string_dest[dsc$b_digits]) and
: 1572      3082 7      (.right_dec eql .diff)) or
: 1573      3083 6      (.right_dec leq .diff)
: 1574      3084 7      then
: 1575      3085 7      begin
: 1576      3086 7      str$dupl_char (.input_string,temp_num_digits,zero);
: 1577      3087 7      pdata_flag = 1;                                ! done - result is zero
: 1578      3088 7      end
: 1579      3089 7
: 1580      3090 6      else
: 1581      3091 7      begin
: 1582      3092 7      buf_ptr = .buf_desc[dsc$a_pointer] + .diff;
: 1583      3093 7      diff = .right_dec - .diff;                                ! move only necessary digits
: 1584      3094 7      +
: 1585      3095 7      Zero out the destination field using the digits
: 1586      3096 7      as the proper number of zero fill characters,
: 1587      3097 7      rather than using the length as found in the
: 1588      3098 7      descriptor, since class SD is a special case.
: 1589      3099 7      -
: 1590      3100 7      str$dupl_char (.input_string,temp_num_digits,zero);
: 1591      3101 7      ch$move (.diff,.buf_ptr,(.input_string[dsc$a_pointer]+(.temp_num_digits-.diff)))
: 1592      3102 7      pdata_flag = 1;
: 1593      3103 7
: 1594      3104 7
: 1595      3105 6      end;
: 1596      3106 6
: 1597      3107 6      end
: 1598      3108 5      else
: 1599      3109 5
: 1600      3110 6      begin
: 1601      3111 6      OK_LEFT = .STRING_DEST [DSC$B_DIGITS] + .STRING_DEST [DSC$B_SCALE] ;
: 1602      3112 6      LENGTH_DIFF = .OK_LEFT - .LEFT_DEC;
: 1603      3113 6      LENGTH_DIFF2 = (.STRING_DEST [DSC$B_DIGITS] - .OK_LEFT) - .RIGHT_DEC;
: 1604      3114 7      IF ( .LENGTH_DIFF LSS 0)
: 1605      3115 6      OR
```

```

1606 3116 7          ( .LENGTH DIFF2 LSS 0)
1607 3117 6          THEN RETURN FAILURE ;
1608 3118 5          end;
1609 3119 4          end;
1610 3120 3          END ;
1611 3121 3
1612 3122 3          [ OTHERWISE ] :
1613 3123 3
1614 3124 3          LIB$STOP ( COB$_INVARG ) ;
1615 3125 3
1616 3126 3          TES ;
1617 3127 3
1618 3128 2          END;
1619 3129 2
1620 3130 2
1621 3131 2      + Figure out the exponent.
1622 3132 2      -
1623 3133 3      IF NOT (.DOT_SEEN)
1624 3134 2      THEN
1625 3135 2          LEFT_DEC = .TEMP_NUM_DIGITS;
1626 3136 2
1627 3137 2      IF .LEFT_DEC EQL 0
1628 3138 2      THEN
1629 3139 2          +
1630 3140 2          - Figure out exponent if all digits are to right of decimal point.
1631 3141 2
1632 3142 2      if .pdata_flag
1633 3143 2      then
1634 3144 2          begin
1635 3145 3          inp = .input_string [dsc$a_pointer];    ! point to re-written data
1636 3146 3          .EXPONENT = 0;
1637 3147 3          INCR GETTER FROM 0 TO (.TEMP_NUM_DIGITS - 1) DO
1638 3148 3              +
1639 3149 3              - Exponent decreases for every leading zero.
1640 3150 3
1641 3151 3              IF .inp [.GETTER] EQL %C'0'
1642 3152 3              THEN
1643 3153 3                  .EXPONENT = ..EXPONENT - 1
1644 3154 3              ELSE
1645 3155 3                  EXITLOOP;
1646 3156 3
1647 3157 3          END
1648 3158 2      ELSE
1649 3159 2          BEGIN
1650 3160 3          .EXPONENT = 0;
1651 3161 3          INCR GETTER FROM 0 TO (.TEMP_NUM_DIGITS - 1) DO
1652 3162 3              +
1653 3163 3              - Exponent decreases for every leading zero.
1654 3164 3
1655 3165 3              IF .BUF [.GETTER] EQL %C'0'
1656 3166 3              THEN
1657 3167 3                  .EXPONENT = ..EXPONENT - 1
1658 3168 3              ELSE
1659 3169 3                  EXITLOOP;
1660 3170 3
1661 3171 2          END
1662 3172 2      ELSE
```

! Data entered too big

! No decimal pt. therefore  
! all digits are left\_dec

```
1663 3173 2
1664 3174 2
1665 3175 2
1666 3176 2
1667 3177 2
1668 3178 2
1669 3179 2
1670 3180 2
1671 3181 2
1672 3182 2
1673 3183 2
1674 3184 2
1675 3185 2
1676 3186 2
1677 3187 2
1678 3188 2
1679 3189 2
1680 3190 2
1681 3191 2
1682 3192 2
1683 3193 2
1684 3194 2
1685 3195 2
1686 3196 2
1687 3197 2
1688 3198 2
1689 3199 2
1690 3200 2
1691 3201 1

      +
      | If all digits are to left of decimal point, exponent is equal
      | to left_dec.
      |
      | .EXPONENT = .LEFT_DEC;
      |
      | +
      | - Move the stripped string into the buffer and adjust the number-of-digits
      |
      | if .pdata_flag eq 0
      | then
      |   begin
      |     CH$MOVE (.TEMP_NUM_DIGITS, .BUF_DESC [DSC$A_POINTER],
      |               .INPUT_STRING [DSC$A_POINTER]);
      |     .NUM_DIGITS = .TEMP_NUM_DIGITS;
      |   end
      | else
      |   if .string_dest[dsc$b_scale] lss 0
      |   then
      |     .num_digits = .temp_num_digits ;
      |
      | +
      | - Free our local string
      |
      | STR$FREE1 DX (BUF_DESC);
      | RETURN SUCCESS ;
      |
      | END;
      |
      | ! end of COB$$$STRIP_BLANKS_SIGN
```

```
00 00 00 30 006AF P.AAD: .BLKB 1
006B0 .ASCII \0\<0><0><0>
ZERO= P.AAD
```

```
OFFC 00000 COB$$$STRIP BLANKS SIGN:
SE      10 1C C2 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 2775
        7E AE D4 00005 SUBL2 #28, SP
        7E 7C 00008 CLRL TEMP_NUM_DIGITS : 2824
        7E 7C 0000A CLRL DIGIT_SEEN
        7E D4 0000C CLRL ZERO_SEEN
        18 AE 7C 0000E CLRL BLANKS_SEEN
        7E D4 00011 CLRL PUTTER
        18 AE D4 00013 CLRL PDATA_FLAG
        2C AE 7C 00016 CLRL LEADING_ZEROES
        6D 027F CF DE 00019 MOVAL 45$, (FP)
        0C BC D5 0001E TSTL @NUM_DIGITS : 2858
        03 12 00021 BNEQ 1$
        026F 31 00023 BRW 43$
        2C AE B4 00026 1$: CLRW BUF_DESC : 2867
        2E AE 0F 90 00029 MOVW #15, BUF_DESC+2 : 2868
        2F AE 02 90 0002D MOVW #2, BUF_DESC+3 : 2869
        30 AE D4 00031 CLRL BUF_DESC+4 : 2870
```

		2C	AE	9F	00034	PUSHAB	BUF_DESC	2871
		0C	AC	DD	00037	PUSHL	NUM_DIGITS	
00000000G	00		02	FB	0003A	CALLS	#1, STR\$GET1 DX	
	58	30	AE	DD	00041	MOVL	BUF_DESC+4, BUF	2876
	59	04	AC	DD	00045	MOVL	INPUT_STRING, R9	2877
	5A	04	A9	9E	00049	MOVAB	4(R9), R10	
	50		6A	DD	0004D	MOVL	(R10), ARG	
14	BC		2B	DD	00050	MOVL	#43, @SIGN_VAL	2878
		04	AE	9F	00054	PUSHAB	BLANKS_SEEN	2884
		0C	AE	9F	00057	PUSHAB	ZERO_SEEN	
		14	AE	9F	0005A	PUSHAB	DOT_SEEN	
		1C	AE	9F	0005D	PUSHAB	DIGIT_SEEN	
		24	AE	9F	00060	PUSHAB	SIGN_SEEN	
		2C	AE	9F	00063	PUSHAB	LEADING_ZEROES	
		34	AE	9F	00066	PUSHAB	PUTTER	
		14	AC	DD	00069	PUSHL	SIGN_VAL	2885
		48	AE	9F	0006C	PUSHAB	TEMP_NUM_DIGITS	2884
		44	AE	9F	0006F	PUSHAB	LEFT_DEC	
		54	AE	9F	00072	PUSHAB	BUF_DESC	
		18	AC	DD	00075	PUSHL	FLAGS	
		0C	BC	DD	00078	PUSHL	@NUM_DIGITS	
			59	DD	0007B	PUSHL	R9	
0000V	CF		0E	FB	0007D	CALLS	#14, COB\$\$\$SCAN_INPUT	
	7C		50	E9	00082	BLBC	R0, 13\$	
	1F	10	AE	E8	00085	BLBS	DIGIT_SEEN, 5\$	2895
	08	14	AE	E8	00089	BLBS	SIGN_SEEN, 2\$	2899
	04	0C	AE	E8	0008D	BLBS	DOT_SEEN, 2\$	
	05	04	AE	E9	00091	BLBC	BLANKS_SEEN, 3\$	
		08	AE	D5	00095	TSTL	ZERO_SEEN	
			67	13	00098	BEQL	13\$	
1C	BE48		3C	90	0009A	MOVB	#48, @PUTTER[BUF]	2902
		1C	AE	D6	0009F	INCL	PUTTER	2903
		28	AE	D6	000A2	INCL	TEMP_NUM_DIGITS	2904
			017D	31	000A5	BRW	31\$	2895
			56	D4	000A8	CLRL	RIGHT_DEC	2912
	05	0C	AE	E8	000AA	BLBS	DOT_SEEN, 6\$	2921
20	AE	28	AE	DD	000AE	MOVL	TEMP_NUM_DIGITS, LEFT_DEC	2922
	54	28	AE	DD	000B3	MOVL	TEMP_NUM_DIGITS, R4	2923
56	54	20	AE	C3	000B7	SUBL3	LEFT_DEC, R4, RIGHT_DEC	
	52		56	DD	000BC	MOVL	RIGHT_DEC, R2	2927
			50	D4	000BF	CLRL	GETTER	
			0C	11	000C1	BRB	8\$	
51	54		50	C3	000C3	SUBL3	GETTER, R4, R1	2928
	30		6148	91	000C7	CMPB	(R1)[BUF], #48	
			06	12	000CB	BNEQ	9\$	
			56	D7	000CD	DECL	RIGHT_DEC	2930
F0	50		52	F3	000CF	AOBLEQ	R2, GETTER, 7\$	2928
	50	08	AC	DD	000D3	MOVL	STRING_DEST, R0	2935
	52		60	3C	000D7	MOVZWL	(R0), DEST_LENGTH	
	51	03	A0	9A	000DA	MOVZBL	3(R0), R1	2937
	01		51	91	000DE	CMPB	R1, #1	2940
			21	12	000E1	BNEQ	14\$	
			56	D5	000E3	TSTL	RIGHT_DEC	2949
			14	15	000E5	BLEQ	12\$	
51	1C	AE	56	C3	000E7	SUBL3	RIGHT_DEC, PUTTER, R1	2951
			51	D7	000EC	DECL	1	
			06	11	000EE	BRB	11\$	

		30	6148	91	000F0	10\$:	CMPB	(1)[BUF], #48	2952
			0B	12	000F4		BNEQ	13\$	
FS		51	1C	AE	F3	000F6	11\$:	AOBLEQ	PUTTER, I, 10\$
		52	20	AE	D1	000FB	12\$:	CMPL	LEFT_DEC, DEST_LENGTH
			A4	15	000FF		BLEQ	4\$	2955
			0195	31	00101	13\$:	BRW	44\$	2956
		09		51	91	00104	14\$:	CMPB	R1, #9
			03	13	00107		BEQL	15\$	2960
			010C	31	00109		BRW	30\$	
		53	09	A0	9E	0010C	15\$:	MOVAB	9(R0), R3
		52	08	A0	98	00110		CVTBL	8(R0), R2
				50	D4	00114		CLRL	R0
				52	D5	00116		TSTL	R2
				6A	15	00118		BLEQ	20\$
				50	D6	0011A		INCL	R0
		5B		63	9A	0011C		MOVZBL	(R3), R11
		5B		52	C0	0011F		ADDL2	R2, R11
	24	AE		5B	D0	00122		MOVL	R11, TOT_DIGITS
				56	D5	00126		TSTL	RIGHT_DEC
				D7	14	00128		BGTR	13\$
	24	AE		54	D1	0012A		CMPL	R4, TOT_DIGITS
				D1	14	0012E		BGTR	13\$
		6E		01	D0	00130		MOVL	#1, PDATA_FLAG
		52		54	D1	00133		CMPL	R4, R2
				03	14	00136		BGTR	16\$
				008A	31	00138		BRW	25\$
			FEBD	CF	9F	0013B	16\$:	PUSHAB	ZERO
			28	AE	9F	0013F		PUSHAB	TOT_DIGITS
				59	DD	00142		PUSHL	R9
	00000000G	00		03	FB	00144		CALLS	#3, STR\$DUPL_CHAR
			18	AE	D5	0014B		TSTL	LEADING_ZEROES
				0F	13	0014E		BEQL	17\$
	57	5B		54	C3	00150		SUBL3	R4, R11, DIFF
		50		63	9A	00154		MOVZBL	(R3), R0
	00 BA47	68		50	28	00157		MOV3	R0, (BUF), @0(R10)[DIFF]
				1F	11	0015D		BRB	19\$
	57	54		52	C3	0015F	17\$:	SUBL3	R2, R4, DIFF
	63	08		00	ED	00163		CMPZV	#0, #8, (R3), DIFF
				07	12	00168		BNEQ	18\$
	00 BA	68		57	28	0016A		MOV3	DIFF, (BUF), @0(R10)
				0D	11	0016F		BRB	19\$
		24	AE	54	C2	00171	18\$:	SUBL2	R4, TOT_DIGITS
	50	6A	24	AE	C1	00175		ADDL3	TOT_DIGITS, (R10), R0
	60	68		57	28	0017A		MOV3	DIFF, (BUF), (R0)
		0C	BC	5B	D0	0017E	19\$:	MOVL	R11, @NUM_DIGITS
				77	11	00182		BRB	27\$
		08		50	E9	00184	20\$:	BLBC	R0, 21\$
		51		63	9A	00187		MOVZBL	(R3), R1
		50		51	D0	0018A		MOVL	R1, OK_LEFT
				07	11	0018D		BRB	22\$
		51		63	9A	0018F	21\$:	MOVZBL	(R3), R1
	50	51		52	C1	00192		ADDL3	R2, R1, OK_LEFT
				65	18	00196	22\$:	BGEQ	28\$
			20	AE	D5	00198		TSTL	LEFT_DEC
				78	14	0019B		BGTR	29\$
				50	D4	0019D		CLRL	OK_LEFT
		50		52	D0	0019F		MOVL	R2, R0

		03	18	001A2	BGEQ	23\$	...
	50	50	CE	001A4	MNEGL	R0, R0	...
	53	50	D0	001A7	23\$: MOVL	R0, OK RIGHT	...
	53	56	D1	001AA	CMPL	RIGHT_DEC, OK RIGHT	3064
		66	14	001AD	BGTR	29\$	...
52	50	51	C3	001AF	SUBL3	R1, R0, DIFF	3079
	6E	01	D0	001B3	MOVL	#1, PDATA_FLAG	3087
	51	56	D1	001B6	CMPL	RIGHT_DEC, R1	3080
		05	12	001B9	BNEQ	24\$	...
	52	56	D1	001BB	CMPL	RIGHT_DEC, DIFF	3081
		05	13	001BE	BEQL	25\$	...
	52	56	D1	001C0	24\$: CMPL	RIGHT_DEC, DIFF	3082
		12	14	001C3	BGTR	26\$	...
		FE33	CF	9F	001C5	25\$: PUSHAB	...
		2C	AE	9F	001C9	PUSHAB	...
			59	DD	001CC	PUSHL	...
	00000000G	00	03	FB	001CE	CALLS	...
			4E	11	001D5	BRB	...
53		52	30	AE	C1	001D7	26\$: ADDL3
52		56	52	C3	001DC	SUBL3	...
			FE18	CF	9F	001E0	PUSHAB
			2C	AE	9F	001E4	PUSHAB
			59	DD	001E7	PUSHL	...
	00000000G	00	03	FB	001E9	CALLS	...
50		28	AE	C3	001F0	SUBL3	...
00 BA40		63	52	28	001F5	MOVC3	...
			28	11	001FB	27\$: BRB	...
50		51	52	C1	001FD	28\$: ADDL3	...
52		50	20	AE	C3	00201	SUBL3
		51	50	C2	00206	SUBL2	...
50		51	56	C3	00209	SUBL3	...
			52	D5	0020D	TSTL	...
			04	19	0020F	BLSS	...
			50	D5	00211	TSTL	...
			10	18	00213	BGEQ	...
			0081	31	00215	29\$: BRW	...
	00000000G	00	8F	DD	00218	30\$: PUSHL	...
		05	01	FB	0021E	CALLS	...
		20	AE	E8	00225	31\$: BLBS	...
		50	28	AE	D0	00229	31\$: MOVL
			10	AC	D0	0022E	32\$: MOVL
			20	AE	D5	00232	TSTL
			35	12	00235	BNEQ	...
53	28	AE	01	C3	00237	SUBL3	...
		18	6E	E9	0023C	BLBC	...
		51	6A	D0	0023F	MOVL	...
			60	D4	00242	CLRL	...
		52	01	CE	00244	MNEGL	...
			08	11	00247	BRB	...
		30	6241	91	00249	33\$: CMPB	...
			21	12	0024D	BNEQ	...
			60	D7	0024F	DECL	...
F4	52	53	F3	00251	34\$: AOBLEQ	R3, GETTER, 33\$	...
		19	11	00255	39\$: BRB	(R0)	...
			60	D4	00257	35\$: CLRL	...
		51	01	CE	00259	MNEGL	...
			08	11	0025C	BRB	...



COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$STRIP\_BLANKS\_SIGN - Pull blanks and sign

G 16  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 47  
(7)

	30		6148	91	0025E	36\$:	CMPB	(GETTER)[BUF], #48	3165	
			0C	12	00262		BNEQ	39\$	3167	
			60	D7	00264		DECL	(R0)	3165	
F4	51		53	F3	00266	37\$:	AOBLEQ	R3, GETTER, 36\$	3142	
			04	11	0026A		BRB	39\$	3177	
	60	20	AE	D0	0026C	38\$:	MOVL	LEFT_DEC, (R0)	3183	
			6E	D5	00270	39\$:	TSTL	PDATA_FLAG	3187	
			09	12	00272		BNEQ	40\$	3188	
00	BA	30	BE	28	AE	28	MOV3	TEMP_NUM_DIGITS, @BUF_DESC+4, @0(R10)	3191	
			09	11	0027B		BRB	41\$	3193	
	50	08	AC	D0	0027D	40\$:	MOVL	STRING_DEST, R0	3198	
		08	A0	95	00281		TSTB	8(R0)	3199	
			05	18	00284		BGEQ	42\$	3201	
	0C	BC	28	AE	D0	00286	41\$:	MOVL	TEMP_NUM_DIGITS, @NUM_DIGITS	2824
		2C	AE	9F	0028B	42\$:	PUSHAB	BUF_DESC		
00000000G	00		01	FB	0028E		CALLS	#1, STR\$FREE1_DX		
	50		01	D0	00295	43\$:	MOVL	#1, R0		
				04	00298		RET			
		50		D4	00299	44\$:	CLRL	R0		
				04	0029B		RET			
				0000	0029C	45\$:	.WORD	Save nothing		
	50	08	AC	D0	0029E		MOVL	8(AP), R0		
	50	04	A0	D0	002A2		MOVL	4(R0), R0		
		F8	A0	9F	002A6		PUSHAB	BUF_DESC		
			01	DD	002A9		PUSHL	#1		
			5E	DD	002AB		PUSHL	SP		
	7E	04	AC	7D	002AD		MOVQ	4(AP), -(SP)		
00000000G	00		03	FB	002B1		CALLS	#3, COB\$\$FREE_STRINGS		
				04	002B8		RET			

; Routine Size: 697 bytes, Routine Base: \_COB\$CODE + 06B4

```
1693 3202 1 %SBTTL 'COB$$ZERO_FILL - Initialize destination'
1694 3203 1 ROUTINE COB$$ZERO_FILL ( STRING_DEST : REF $STR$DESCRIPTOR
1695 3204 1                                     ! Destination for input
1696 3205 1                                     ) : NOVALUE =
1697 3206 1
1698 3207 1 !++
1699 3208 1 ! FUNCTIONAL DESCRIPTION:
1700 3209 1 !
1701 3210 1 !     This routine will initialize STRING_DEST to zeroes before the input
1702 3211 1 !     data is copied to it.
1703 3212 1 !
1704 3213 1 ! FORMAL PARAMETERS:
1705 3214 1 !
1706 3215 1 !     STRING_DEST.mt.ds    Address of descriptor to receive the read input.
1707 3216 1 !
1708 3217 1 ! IMPLICIT INPUTS:
1709 3218 1 !
1710 3219 1 !     NONE
1711 3220 1 !
1712 3221 1 ! IMPLICIT OUTPUTS:
1713 3222 1 !
1714 3223 1 !     NONE
1715 3224 1 !
1716 3225 1 ! ROUTINE VALUE:
1717 3226 1 !
1718 3227 1 !     NONE
1719 3228 1 !
1720 3229 1 ! SIDE EFFECTS:
1721 3230 1 !
1722 3231 1 ! --
1723 3232 1
1724 3233 2     BEGIN
1725 3234 2
1726 3235 2     LOCAL
1727 3236 2         SIGN          : BYTE,
1728 3237 2         ZERO          : BYTE,
1729 3238 2         DEST_PTR      :
1730 3239 2         DEST_LENGTH    :
1731 3240 2
1732 3241 2     LITERAL
1733 3242 2         ZERO_0    = 48,
1734 3243 2         POS_SIGN  = 43,
1735 3244 2         POSZEROP = 123 ;
1736 3245 2
1737 3246 2
1738 3247 2     ZERO = ZERO_0 ;
1739 3248 2     DEST_PTR = .STRING_DEST [DSC$A_POINTER] ;
1740 3249 2     DEST_LENGTH = .STRING_DEST [DSC$W_LENGTH] ;
1741 3250 2
1742 3251 2     !+
1743 3252 2     ! Zero fill, then handle sign correctly.
1744 3253 2     !-
1745 3254 2
1746 3255 2     STR$DUPL_CHAR ( .STRING_DEST, DEST_LENGTH, ZERO ) ;
1747 3256 2
1748 3257 2     CASE .STRING_DEST [DSC$B_DTYPE] FROM DSC$K_DTYPE_NU TO DSC$K_DTYPE_NRO
1749 3258 2     OF
```

```

: 1750      3259 2      SET
: 1751      3260 2
: 1752      3261 2      [DSC$K_DTYPE_NU]:                ! Numeric unsigned
: 1753      3262 2      +
: 1754      3263 2      | Move all zeroes to STRING_DEST
: 1755      3264 2      -
: 1756      3265 2      0 ;                                ! No further action
: 1757      3266 2
: 1758      3267 2      [DSC$K_DTYPE_NL]:                ! Numeric left separate
: 1759      3268 2      +
: 1760      3269 2      | Move sign then all zeroes to STRING_DEST
: 1761      3270 2      -
: 1762      3271 2
: 1763      3272 2      BEGIN
: 1764      3273 3
: 1765      3274 3      SIGN = POS_SIGN ;
: 1766      3275 3      CH$MOVE (1, SIGN, .STRING_DEST [DSC$A_POINTER]);
: 1767      3276 3
: 1768      3277 2      END;
: 1769      3278 2
: 1770      3279 2      [DSC$K_DTYPE_NR]:                ! Numeric right separate
: 1771      3280 2      +
: 1772      3281 2      | Move all zeroes followed by sign to STRING_DEST
: 1773      3282 2      -
: 1774      3283 2
: 1775      3284 3      BEGIN
: 1776      3285 3
: 1777      3286 3      SIGN = POS_SIGN ;
: 1778      3287 3      CH$MOVE (1, SIGN, (.DEST_PTR + (.DEST_LENGTH - 1)) ) ;
: 1779      3288 3
: 1780      3289 2      END;
: 1781      3290 2
: 1782      3291 2      [DSC$K_DTYPE_NLO]:                ! Numeric left overpunched
: 1783      3292 2      +
: 1784      3293 2      | Move all zeroes to STRING_DEST
: 1785      3294 2      | First digit has overpunch sign (positive)
: 1786      3295 2      -
: 1787      3296 2
: 1788      3297 3      BEGIN
: 1789      3298 3
: 1790      3299 3      SIGN = POSZEROP ;
: 1791      3300 3      CH$MOVE (1, SIGN, .STRING_DEST[DSC$A_POINTER]);
: 1792      3301 3
: 1793      3302 2      END;
: 1794      3303 2
: 1795      3304 2      [DSC$K_DTYPE_NRO]:                ! Numeric right overpunched
: 1796      3305 2      +
: 1797      3306 2      | Move all zeroes to STRING_DEST
: 1798      3307 2      | Last digit has overpunch sign (positive)
: 1799      3308 2      -
: 1800      3309 2
: 1801      3310 3      BEGIN
: 1802      3311 3
: 1803      3312 3      SIGN = POSZEROP ;
: 1804      3313 3      CH$MOVE (1, SIGN, (.DEST_PTR + (.DEST_LENGTH - 1)) );
: 1805      3314 3
: 1806      3315 2      END;
```

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$ZERO\_FILL - Initialize destination

J 16  
13-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 50  
(8)

: 1807  
: 1808  
: 1809  
: 1810

3316 2  
3317 2  
3318 2  
3319 1

TES;

END ;

! End of COB\$\$ZERO\_FILL

				000C 00000 COB\$\$ZERO_FILL:				
		5E	08	C2	00002	.WORD	Save R2,R3	: 3203
		6E	30	90	00005	SUBL2	#8, SP	: 3204
		52	04	AC	D0 00008	MOVB	#48, ZERO	: 3247
		53	04	A2	D0 0000C	MOVL	STRING_DEST, R2	: 3248
		AE	62	3C	00010	MOVL	4(R2), DEST_PTR	: 3249
			5E	DD	00014	MOVZWL	(R2), DEST_LENGTH	: 3255
			08	AE	9F 00016	PUSHL	S3	: 3256
			52	DD	00019	PUSHAB	DEST_LENGTH	: 3257
			03	FB	0001B	PUSHL	R2	: 3258
			02	A2	8F 00022	CALLS	#3, STR\$DUPL_CHAR	: 3259
0010	04	00000000G	00			CASEB	2(R2), #15, #4	: 3260
	0015	000B	002B		00027	.WORD	8\$-1\$,-	: 3261
			001E		0002F		2\$-1\$,-	: 3262
							4\$-1\$,-	: 3263
							3\$-1\$,-	: 3264
							6\$-1\$	: 3265
		50	2B	90	00031	RET		: 3274
			09	11	00032	MOVB	#43, SIGN	: 3275
		50	2B	90	00035	BRB	5\$	: 3286
			0D	11	00037	MOVB	#43, SIGN	: 3287
		50	7B	8F	90 0003A	BRB	7\$	: 3299
	04	B2	50	90	0003C	MOVB	#123, SIGN	: 3300
				90	00040	MOVB	SIGN, 24(R2)	: 3257
				04	00044	RET		: 3312
	51	50	7B	8F	90 00045	MOVB	#123, SIGN	: 3313
		53	04	AE	C1 00049	ADDL3	DEST_LENGTH, DEST_PTR, R1	: 3319
		FF	A1	50	90 0004E	MOVB	SIGN, -1(R1)	: 3319
				04	00052	RET		: 3319

; Routine Size: 83 bytes, Routine Base: \_COB\$CODE + 096D

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$VERIFY\_FL\_RANGE - Verify Float Pt range

K 16  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 51  
(9)

```
1812 3320 1 %SBTTL 'COB$$VERIFY_FL_RANGE - Verify Float Pt range'
1813 3321 1 ROUTINE COB$$VERIFY_FL_RANGE (
1814 3322 1     TEMP_PUT_HERE : REF BLOCK [8, BYTE], ! # to scan
1815 3323 1     CHARS_READ,      ! # of chars in TEMP_PUT_HERE
1816 3324 1     MAX          ! # of significant digits allowed
1817 3325 1 ) =
1818 3326 1
1819 3327 1 ++
1820 3328 1 FUNCTIONAL DESCRIPTION:
1821 3329 1
1822 3330 1     Check range of Floating and Double Floating Point input data.
1823 3331 1     Do nothing about errors in this routine, return to calling routine.
1824 3332 1
1825 3333 1 FORMAL PARAMETERS:
1826 3334 1
1827 3335 1     TEMP_PUT_HERE.rt.dx      Input data to be verified.
1828 3336 1
1829 3337 1     CHARS_READ.rlu.v        Number of input characters.
1830 3338 1
1831 3339 1     MAX.rlu.v              Number of significant digits allowed in
1832 3340 1                          mantissa of E notation representation.
1833 3341 1                          7 for Floating Point
1834 3342 1                          16 for Double Floating Point
1835 3343 1
1836 3344 1 IMPLICIT INPUTS:
1837 3345 1
1838 3346 1     NONE
1839 3347 1
1840 3348 1 IMPLICIT OUTPUTS:
1841 3349 1
1842 3350 1     NONE
1843 3351 1
1844 3352 1 ROUTINE VALUE:
1845 3353 1
1846 3354 1     1 = SUCCESS
1847 3355 1     0 = FAILURE
1848 3356 1
1849 3357 1 SIDE EFFECTS:
1850 3358 1
1851 3359 1     NONE
1852 3360 1 --
1853 3361 2 BEGIN
1854 3362 2
1855 3363 2     +
1856 3364 2     This routine counts the significant digits (1-9) and significant zeroes
1857 3365 2     of data that is input to either a floating point or a double floating
1858 3366 2     point data item.  Some zeroes can be ignored.
1859 3367 2
1860 3368 2     000000000012.340000000000000000
1861 3369 2     \-----/ \-----/
1862 3370 2     ignore      ignore
1863 3371 2
1864 3372 2     000.0000000000000000000000000012345
1865 3373 2     \-----/ \-----/
1866 3374 2     ignore      do not ignore
1867 3375 2
1868 3376 2
```

COBSACCECV  
1-001

COBSACCECV - ACCEPT Conversion routines  
COBS\$VERIFY\_FL\_RANGE - Verify Float Pt range

L 16  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 52  
(9)

```
1869 3377 2 1234000000000000000000000000000000.000
1870 3378 2
1871 3379 2
1872 3380 2
1873 3381 2
1874 3382 2
1875 3383 2
1876 3384 2 LOCAL
1877 3385 2 PUT BUF : REF VECTOR [1100, BYTE],
1878 3386 2 COUNT : INITIAL (0), Count for TEMP_PUT_HERE
1879 3387 2 DOT_SEEN : INITIAL (0), =1 Decimal point seen
1880 3388 2 DIGIT_SEEN : INITIAL (0), =1 At least 1 digit seen
1881 3389 2 SIGN_SEEN : INITIAL (0), =1 Sign seen
1882 3390 2 E_SEEN : INITIAL (0), =1 E/e of exponent seen
1883 3391 2 R_SIGNIF : INITIAL (0), Significant digits to
1884 3392 2 right of decimal point
1885 3393 2 L_SIGNIF : INITIAL (0), Significant digits to
1886 3394 2 left of decimal point
1887 3395 2 R_ZERO : INITIAL (0), Significant zeroes to
1888 3396 2 right of decimal point
1889 3397 2 ! Calculated after incr loop .00...00123
1890 3398 2 L_ZERO : INITIAL (0), Significant zeroes to
1891 3399 2 left of decimal point
1892 3400 2 ! Calculated in incr loop 1200...0.0
1893 3401 2
1894 3402 2 PUT_BUF = .TEMP_PUT_HERE [DSC$A_POINTER] ;
1895 3403 2
1896 3404 2 INCR X FROM 0 TO .CHARS_READ - 1 DO
1897 3405 2 BEGIN ! Begin INCR loop
1898 3406 2
1899 3407 2 Count significant digits and significant zeroes.
1900 3408 2
1901 3409 2 SELECT ONE .PUT_BUF [.X] OF
1902 3410 2 SET
1903 3411 2 [ %C'1' TO %C'9' ] :
1904 3412 2
1905 3413 2 Count significant digits to the left
1906 3414 2 or right of the decimal point.
1907 3415 2
1908 3416 2
1909 3417 2 BEGIN
1910 3418 2 DIGIT_SEEN = 1 ;
1911 3419 2 IF .DOT_SEEN
1912 3420 2 THEN
1913 3421 2 R_SIGNIF = .R_SIGNIF + 1
1914 3422 2 ELSE
1915 3423 2 L_SIGNIF = .L_SIGNIF + 1 ;
1916 3424 2 COUNT = .COUNT + 1 ;
1917 3425 2 END ;
1918 3426 2
1919 3427 2 [ %C'0' ] :
1920 3428 2
1921 3429 2 Count zeroes after DIGIT_SEEN
1922 3430 2 and/or after DOT_SEEN.
1923 3431 2
1924 3432 2
1925 3433 2
```

COBSACCECV  
1-001

COBSACCECV - ACCEPT Conversion routines  
COBS\$VERIFY\_FL\_RANGE - Verify Float Pt range

M 16  
15-Sep-1984 23:49:06 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:10:22 [COBRTL.SRC]COBACCECV.B32;1

Page 53  
(9)

```

: 1926 3434 4
: 1927 3435 4
: 1928 3436 4
: 1929 3437 4
: 1930 3438 4
: 1931 3439 4
: 1932 3440 4
: 1933 3441 4
: 1934 3442 4
: 1935 3443 4
: 1936 3444 4
: 1937 3445 4
: 1938 3446 4
: 1939 3447 4
: 1940 3448 4
: 1941 3449 4
: 1942 3450 4
: 1943 3451 3
: 1944 3452 3
: 1945 3453 3
: 1946 3454 3
: 1947 3455 3
: 1948 3456 3
: 1949 3457 3
: 1950 3458 3
: 1951 3459 3
: 1952 3460 4
: 1953 3461 4
: 1954 3462 4
: 1955 3463 4
: 1956 3464 3
: 1957 3465 3
: 1958 3466 3
: 1959 3467 3
: 1960 3468 3
: 1961 3469 4
: 1962 3470 4
: 1963 3471 4
: 1964 3472 3
: 1965 3473 3
: 1966 3474 3
: 1967 3475 3
: 1968 3476 3
: 1969 3477 3
: 1970 3478 3
: 1971 3479 3
: 1972 3480 3
: 1973 3481 3
: 1974 3482 3
: 1975 3483 3
: 1976 3484 3
: 1977 3485 3
: 1978 3486 4
: 1979 3487 4
: 1980 3488 4
: 1981 3489 3
: 1982 3490 3

```

```

BEGIN
IF .DIGIT_SEEN
THEN
  IF .DOT_SEEN
  THEN
    R_SIGNIF = .R_SIGNIF + 1
  ELSE
    L_SIGNIF = .L_SIGNIF + 1
  ELSE
    +
    | Count zeroes after decimal point,
    | but before significant digits
    -
    IF .DOT_SEEN
    THEN
      R_ZERO = .R_ZERO + 1 ;
    COUNT = .COUNT + 1 ;
    END ;
[ %C'-' , %C'+' ] :
  +
  | Only one sign is valid
  -
  IF .SIGN_SEEN EQL 0
  THEN
    BEGIN
      SIGN_SEEN = 1 ;
      COUNT = .COUNT + 1 ;
    END
  ELSE
    RETURN 0 ;
[ %C'.' , %C',' ] :
  BEGIN
    DOT_SEEN = 1 ;
    COUNT = .COUNT + 1 ;
    END ;
[ %C' ' ] :
  +
  | Spaces are allowed
  -
  COUNT = .COUNT + 1 ;
[ %C'E' , %C'e' ] :
  +
  | Don't check range of exponent, leave
  | that for OTS$CVT_I_F/D
  -
  BEGIN
    E_SEEN = 1 ;
    EXITLOOP ;
  END ;

```

[ OTHERWISE ] :

RETURN 0 ;

YES :

\* If maximum significant digits allowed has already  
been reached, or all is left is the exponent  
- pull out of loop.

IF (.L\_SIGNIF + .R\_SIGNIF EQL .MAX) OR  
(.E\_SEEN)

THEN EXITLOOP ;

END ;

! End INCR loop

\* Make sure all remaining digits (if any)  
are zeroes  
Count zeroes after significant digits, but  
before decimal point.

IF .COUNT LSS .CHARS\_READ  
THEN

\* All input characters have not yet been processed.

BEGIN

INCR Y FROM .COUNT TO .CHARS\_READ - 1 DO  
BEGIN

SELECT ONE .PUT\_BUF [Y] OF  
SET

[ %C'E', %C'e' ] :

\* Don't check range of exponent,  
leave that for OTSSCVT\_T.F/D

EXITLOOP ;

[ %C'O' ] :

\* Count zeroes to left  
of decimal point.

IF .DOT\_SEEN EQL 0

THEN

L\_ZERO = .L\_ZERO + 1 .

[ %C', %C', %C'-', %C'+', %C' ' ]

\* Of no consequence here



[illegible]

OFFC 00000 COBSSVERIFY FL-RANGE:					
				.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
		52	D4 00002	CLRL	COUNT
		55	7C 00004	CLRQ	SIGN SEEN
		54	D4 00006	CLRL	E_SEEN
		58	7C 00008	CLRQ	R-SIGNIF
		57	D4 0000A	CLRL	L-SIGNIF
		5A	7C 0000C	CLRQ	L_ZERO
50	04	AC	D0 0000E	MOVL	TEMP PUT HERE, R0
51	04	A0	D0 00012	MOVL	4(ROT, PUT_BUF
53		01	CE 00016	MNEGL	#1, X
		76	11 00019	BRB	16\$
50	63	41	9A 0001B	MOVZBL	(X)[PUT_BUF], R0
31		50	91 0001F	CMPB	R0, #49
		0A	1F 00022	BLSSU	2\$
39		50	91 00024	CMPB	R0, #57
		05	1A 00027	BGTRU	2\$

56	01	D0	00029	MOVL	#1, DIGIT_SEEN	3419		
	08	11	0002C	BRB	38	3420		
30	50	91	0002E	28: CMPB	RO, #48	3428		
	15	12	00031	BNEQ	68			
08	56	E9	00033	BLBC	DIGIT_SEEN, 58	3435		
04	59	E9	00036	38: BLBC	DOT_SEEN, 48	3437		
	58	D6	00039	INCL	R_SIGNIF	3439		
	32	11	0003B	BRB	178			
	57	D6	0003D	48: INCL	L_SIGNIF	3441		
	2E	11	0003F	BRB	178	3437		
23	59	E9	00041	58: BLBC	DOT_SEEN, 118	3447		
	58	D6	00044	INCL	R_ZERO	3449		
	27	11	00046	BRB	178	3450		
28	50	91	00048	68: CMPB	RO, #43	3453		
	05	13	0004B	BEQL	78			
20	50	91	0004D	CMPB	RO, #45			
	09	12	00050	BNEQ	88			
	55	D5	00052	78: TSTL	SIGN_SEEN	3458		
	27	12	00054	BNEQ	138			
55	01	D0	00056	MOVL	#1, SIGN_SEEN	3461		
	14	11	00059	BRB	118	3462		
2C	50	91	0005B	88: CMPB	RO, #44	3467		
	05	13	0005E	BEQL	98			
2E	50	91	00060	CMPB	RO, #46			
	05	12	00063	BNEQ	108			
59	01	D0	00065	98: MOVL	#1, DOT_SEEN	3470		
	05	11	00068	BRB	118	3471		
20	50	91	0006A	108: CMPB	RO, #32	3474		
	04	12	0006D	BNEQ	128			
	52	D6	0006F	118: INCL	COUNT	3478		
	11	11	00071	BRB	158			
45	8F	50	91	00073	128: CMPB	RO, #69	3480	
	06	13	00077	BEQL	148			
65	8F	50	91	00079	CMPB	RO, #101		
	68	12	0007D	138: BNEQ	228			
54	01	D0	0007F	148: MOVL	#1, E_SEEN	3487		
	12	11	00082	BRB	178	3486		
50	57	58	C1	00084	158: ADDL3	R_SIGNIF, L_SIGNIF, RO	3503	
	0C	AC	50	D1	00088	CMPL	RO, MAX	
		08	13	0008C	BEQL	178		
	05	54	E8	0008E	BLBS	E_SEEN, 178	3504	
85	53	08	AC	F2	00091	168: AOBLS	CHARS_READ, X, 18	3404
	08	AC	52	D1	00096	178: CMPL	COUNT, CHARS_READ	3516
			35	18	0009A	BGEQ	218	
			52	D7	0009C	DECL	Y	3522
			2C	11	0009E	BRB	208	
	50		6241	9A	000A0	188: MOVZBL	(Y)[PUT_BUF], RO	3525
45	8F		50	91	000A4	CMPB	RO, #69	3528
			27	13	000A8	BEQL	218	
65	8F		50	91	000AA	CMPB	RO, #101	
			21	13	000AE	BEQL	218	
	30		50	91	000B0	CMPB	RO, #48	3536
			08	12	000B3	BNEQ	198	
			59	D5	000B5	TSTL	DOT_SEEN	3541
			13	12	000B7	BNEQ	208	
			5A	D6	000B9	INCL	L_ZERO	3543
			0F	11	000BB	BRB	208	3541

COBSACCECV  
1-001

COBSACCECV - ACCEPT Conversion routines  
COBS\$VERIFY\_FL\_RANGE - Verify Float Pt range

E 1  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBSACCECV.B32;1

Page 57  
(9)

	20		50	91	000BD	19\$:	CMPB	R0	#32		: 3545
			0A	13	000C0		BEQL	20\$			: :
	2B		50	91	000C2		CMPB	R0	#43		: :
			20	1F	000C5		BLSSU	22\$			: :
	2E		50	91	000C7		CMPB	R0	#46		: :
			1B	1A	000CA		BGTRJ	22\$			: :
CF	52	0B	AC	F2	000CC	20\$:	AOBLSS	CHARS READ, Y, 18\$			: 3522
50	5B		5B	C1	000D1	21\$:	ADDL3	R_ZERO, R_SIGNIF, R0			: 3576
	3B		50	D1	000D5		CMPL	R0	#56		: :
			0D	14	000D8		BGTR	22\$			: :
50	57		5A	C1	000DA		ADDL3	L_ZERO, L_SIGNIF, R0			: 3577
	3B		50	D1	000DE		CMPL	R0	#56		: :
			04	14	000E1		BGTR	22\$			: :
	50		01	D0	000E3		MOVL	#1, R0			: 3581
			04	000E6			RET				: :
			50	D4	000E7	22\$:	CLRL	R0			: 3582
			04	000E9			RET				: :

; Routine Size: 234 bytes, Routine Base: \_COB\$CODE + 09C0

```
2076 3583 1 %SBTTL 'COBSSCAN_INPUT - Scan the input string'
2077 3584 1 ROUTINE COBSSCAN_INPUT ( ARG_DESC : REF $STR$DESCRIPTOR,
2078 3585 1 CHARS_READ,
2079 3586 1 FLAGS,
2080 3587 1 BUF_DESC : REF $STR$DESCRIPTOR,
2081 3588 1 LEFT_DEC,
2082 3589 1 NUM_DIGITS,
2083 3590 1 SIGN_VAL,
2084 3591 1 PUTTER,
2085 3592 1 LEADING_ZEROES,
2086 3593 1 SIGN_SEEN,
2087 3594 1 DIGIT_SEEN,
2088 3595 1 DOT_SEEN,
2089 3596 1 ZERO_SEEN,
2090 3597 1 BLANKS_SEEN
2091 3598 1 ) : =
2092 3599 1
2093 3600 1 **
2094 3601 1 FUNCTIONAL DESCRIPTION:
2095 3602 1
2096 3603 1 Scan the input number, put result in BUF.
2097 3604 1
2098 3605 1 FORMAL PARAMETERS:
2099 3606 1
2100 3607 1 ARG_DESC.rt.dx The numeric string to scan
2101 3608 1 CHARS_READ.rlu.v Number of input characters read
2102 3609 1 FLAGS.rlu.v Screen enhancement flag
2103 3610 1 BUF_DESC.rt.dx Deposit for scanned input
2104 3611 1 LEFT_DEC.ml.r # of digits to left of decimal point
2105 3612 1 NUM_DIGITS.ml.r # of digits in ARG_DESC
2106 3613 1 SIGN_VAL.ml.r Temp to hold sign
2107 3614 1 PUTTER.ml.r Counts position in buffer BUF_DESC
2108 3615 1 LEADING_ZEROES.ml.r # of leading zeroes
2109 3616 1 SIGN_SEEN.ml.r = 1 if '+' or '-' scanned
2110 3617 1 DIGIT_SEEN.ml.r = 1 if a digit 0-9 was scanned
2111 3618 1 DOT_SEEN.ml.r = 1 if '.' or ',' scanned
2112 3619 1 ZERO_SEEN.ml.r = 1 if digit 0 scanned
2113 3620 1 BLANKS_SEEN.ml.r = 1 if trailing blanks scanned
2114 3621 1
2115 3622 1 IMPLICIT INPUTS:
2116 3623 1
2117 3624 1 NONE
2118 3625 1
2119 3626 1 IMPLICIT OUTPUTS:
2120 3627 1
2121 3628 1 NONE
2122 3629 1
2123 3630 1 ROUTINE VALUE:
2124 3631 1 COMPLETION CODES:
2125 3632 1
2126 3633 1 0 = Failure, 1 = Success
2127 3634 1
2128 3635 1 SIDE EFFECTS:
2129 3636 1
2130 3637 1 NONE
2131 3638 1 --
2132 3639 1
```

```
2133 3640 2 BEGIN
2134 3641 LOCAL
2135 3642 BUF : REF VECTOR [1100, BYTE], . Addresses result
2136 3643 ARG : REF VECTOR [1100, BYTE]; . Addresses source
2137 3644
2138 3645 BUF = .BUF_DESC [DSC$A_POINTER];
2139 3646 ARG = .ARG_DESC [DSC$A_POINTER];
2140 3647
2141 3648 !+
2142 3649 Scan Input, put result in B.F.
2143 3650 !-
2144 3651
2145 3652 INCR GETTER FROM 0 TO (.CHARS_READ - 1) DO
2146 3653
2147 3654 SELECTONE .ARG [.GETTER] OF
2148 3655 SET
2149 3656
2150 3657 [XC'0' TO XC'9'] :
2151 3658
2152 3659 BEGIN ! Decimal digit
2153 3660 IF ( .ARG [.GETTER] NEQ XC'0' ) OR ( ..DOT_SEEN EQL 1 ) OR
2154 3661 ((.ARG [.GETTER] EQL XC'0' ) AND ..DIGIT_SEEN EQL 1)
2155 3662 THEN
2156 3663 BEGIN
2157 3664 !+
2158 3665 This is not a leading zero
2159 3666 !-
2160 3667 IF ..BLANKS_SEEN ! Ensure no imbedded blanks
2161 3668 THEN RETURN 0;
2162 3669
2163 3670 .DIGIT_SEEN = 1;
2164 3671 BUF [.PUTTER] = .ARG [.GETTER];
2165 3672 .PUTTER = ..PUTTER + 1;
2166 3673
2167 3674 .NUM_DIGITS = ..NUM_DIGITS + 1 ;
2168 3675 END
2169 3676 ELSE
2170 3677 BEGIN
2171 3678 .LEADING_ZEROES = ..LEADING_ZEROES + 1;
2172 3679 .ZERO_SEEN = 1 ; ! 00. is valid - dot_seen and
2173 3680 ! zero_seen
2174 3681 END;
2175 3682 END;
2176 3683
2177 3684 [XC'+' , XC'-'] :
2178 3685
2179 3686 BEGIN ! Plus or minus sign
2180 3687 IF ( ..SIGN_SEEN ) THEN RETURN 0 ;
2181 3688 IF .GETTER NEQ .CHARS_READ - 1 ! Ensure no imbedded signs
2182 3689 THEN
2183 3690 IF (( ..DIGIT_SEEN ) AND ( .ARG [.GETTER + 1] NEQ XC' ' ))
2184 3691 THEN RETURN 0;
2185 3692
2186 3693 .SIGN_SEEN = 1;
2187 3694 .SIGN_VAL = .ARG [.GETTER];
2188 3695 END;
2189 3696
```

```
2190 3697 2      [XC'.'] :
2191 3698 2
2192 3699 2      BEGIN
2193 3700 2          IF ( ..DOT_SEEN ) THEN RETURN 0;      ! Decimal point
2194 3701 2          +
2195 3702 2          ! Is decimal point a valid character - look at bit 6 of FLAGS
2196 3703 2          -
2197 3704 2          IF ( .FLAGS AND V_DEC_PT ) NEQ 0
2198 3705 2          THEN
2199 3706 2              RETURN 0      ! Decimal point is illegal
2200 3707 2          ELSE
2201 3708 2              BEGIN
2202 3709 2                  .DOT_SEEN = 1;
2203 3710 2                  .LEFT_DEC = ..NUM_DIGITS ;      ! Count for validating size of
2204 3711 2                  END ;      ! entered data.  NUM_DIGITS
2205 3712 2                  ! calculated below.
2206 3713 2              END;
2207 3714 2      [XC',' ] :
2208 3715 2      BEGIN
2209 3716 2          IF ( ..DOT_SEEN ) THEN RETURN 0;      ! Decimal point is Comma
2210 3717 2          +
2211 3718 2          ! Is comma a valid character - look at bit 6 of FLAGS
2212 3719 2          -
2213 3720 2          IF ( .FLAGS AND V_DEC_PT ) NEQ 0
2214 3721 2          THEN
2215 3722 2              BEGIN
2216 3723 2                  ! Comma is an illegal character
2217 3724 2                  .DOT_SEEN = 1;
2218 3725 2                  .LEFT_DEC = ..NUM_DIGITS ;      ! Count for validating size of
2219 3726 2                  END      ! entered data.  NUM_DIGITS
2220 3727 2                  ! calculated below.
2221 3728 2              ELSE
2222 3729 2                  RETURN 0 ;
2223 3730 2              END;
2224 3731 2      [XC' ' ] :
2225 3732 2      BEGIN
2226 3733 2          ! Blank, better be leading or trailing.
2227 3734 2          IF ( ..SIGN_SEEN OR ..DIGIT_SEEN OR ..DOT_SEEN )
2228 3735 2          THEN .BLANKS_SEEN = 1;
2229 3736 2          END;
2230 3737 2
2231 3738 2      [OTHERWISE] :      ! reprompt by passing back a routine value of 0
2232 3739 2
2233 3740 2      RETURN 0 ;
2234 3741 2      TES;
2235 3742 2
2236 3743 2      RETURN 1 ;
2237 3744 1      END ;      ! End COB$$$SCAN_INPUT
```

003C 00000 COB\$\$\$SCAN INPUT:

```
50      10  AC  DO 00002      .WORD  Save R2,R3,R4,R5
55      04  AO  DO 00006      MOVL   BUF_DESC, R0
                                MOVL   4(R0), BUF
```

```
: 3584
: 3645
:
```

	5C	04	AC	D0	0000A	MOVL	ARG DESC, R0	3646	
	52	04	A0	D0	0000E	MOVL	4(R0), ARG		
	54		01	CE	00012	MNEGL	#1, GETTER	3654	
			75	11	00015	BRB	8\$		
	50		0442	9A	00017	1\$:	MOVZBL (GETTER)[ARG], R0		
	30		50	91	0001B	CMPB	R0, #48	3657	
			41	1F	0001E	BLSSU	5\$		
	39		50	91	00020	CMPB	R0, #57		
			3C	1A	00023	BGTRU	5\$		
	30		50	91	00025	CMPB	R0, #48	3660	
			11	12	00028	BNEQ	2\$		
	01	30	BC	D1	0002A	CMPL	@DOT_SEEN, #1		
			0B	13	0002E	BEQL	2\$		
	30		50	91	00030	CMPB	R0, #48	3661	
			23	12	00033	BNEQ	4\$		
	01	2C	BC	D1	00035	CMPL	@DIGIT_SEEN, #1		
			1D	12	00039	BNEQ	4\$		
	03	38	BC	E9	0003B	2\$:	BLBC @BLANKS_SEEN, 3\$	3667	
			0098	31	0003F	BRW	17\$		
	2C	BC	01	D0	00042	3\$:	MOVL #1, @DIGIT_SEEN	3670	
			51	AC	00046	MOVL	PUTTER, R1	3671	
			53	61	0004A	MOVL	(R1), R3		
	6345		50	90	0004D	MOVB	R0, (R3)[BUF]		
			61	D6	00051	INCL	(R1)	3672	
			18	BC	D6	00053	INCL	@NUM_DIGITS	3674
			74	11	00056	BRB	14\$	3660	
			24	BC	D6	00058	4\$:	INCL @LEADING_ZEROES	3678
	34	BC	01	D0	0005B	MOVL	#1, @ZERO_SEEN	3679	
			6B	11	0005F	BRB	14\$	3654	
			2B	50	91	00061	5\$:	CMPB R0, #43	3684
				05	13	00064	BEQL	6\$	
			2D	50	91	00066	CMPB	R0, #45	
				23	12	00069	BNEQ	9\$	
			6B	28	BC	E8	6\$:	BLBS @SIGN_SEEN, 17\$	3687
51			08	AC	01	C3	SUBL3	#1, CHARS_READ, R1	3688
				51	54	D1	CMPL	GETTER, RT	
					0B	13	BEQL	7\$	
			07	BC	E9	00079	BLBC	@DIGIT_SEEN, 7\$	3690
			20	01	A442	91	CMPL	1(GETTER)[ARG], #32	
					56	12	BNEQ	17\$	
			28	BC	01	D0	7\$:	MOVL #1, @SIGN_SEEN	3693
			1C	BC	50	D0	MOVL	R0, @SIGN_VAL	3694
					3E	11	BRB	14\$	3654
			2E	50	91	0008E	9\$:	CMPB R0, #46	3697
				0B	12	00091	BNEQ	10\$	
			43	30	BC	E8	BLBS	@DOT_SEEN, 17\$	3700
3E			0C	AC	06	E0	BBS	#6, FLAGS, 17\$	3704
					0E	11	BRB	11\$	3709
			2C	50	91	0009E	10\$:	CMPB R0, #44	3714
				14	12	000A1	BNEQ	12\$	
			33	30	BC	E8	BLBS	@DOT_SEEN, 17\$	3717
2E			0C	AC	06	E1	BBS	#6, FLAGS, 17\$	3721
			30	BC	01	D0	11\$:	MOVL #1, @DOT_SEEN	3724
			14	BC	18	D0	MOVL	@NUM_DIGITS, @LEFT_DEC	3725
					15	11	BRB	14\$	3721
			20	50	91	000B7	12\$:	CMPB R0, #32	3731
				1E	12	000BA	BNEQ	17\$	

COB\$ACCECV  
1-001

COB\$ACCECV - ACCEPT Conversion routines  
COB\$\$SCAN\_INPUT - Scan the input string

J 1  
15-Sep-1984 23:49:06  
14-Sep-1984 12:10:22

VAX-11 Bliss-32 V4.0-742  
[COBRTL.SRC]COBACCECV.B32;1

Page 62  
(10)

	08	28	BC	E8	000BC		BLBS	@SIGN SEEN, 13\$	:	3734
	04	2C	BC	E8	000C0		BLBS	@DIGIT SEEN, 13\$	:	
	04	30	BC	E9	000C4		BLBC	@DOT SEEN, 14\$	:	
02	38	BC	01	D0	000C8	13\$:	MOVL	#1, @BLANKS SEEN	:	3735
	54	08	AC	F2	000CC	14\$:	AOBLSS	CHARS_READ, -GETTER, 15\$	:	3654
			03	11	000D1		BRB	16\$	:	
			FF41	31	000D3	15\$:	BRW	1\$	:	
	50		01	D0	000D6	16\$:	MOVL	#1, R0	:	3743
			04	000D9			RET		:	
			50	D4	000DA	17\$:	CLRL	R0	:	3744
			04	000DC			RET		:	

; Routine Size: 221 bytes, Routine Base: \_COB\$CODE + 0AAA

: 2238	3745	1		
: 2239	3746	1	END	! End of module COB\$ACCECV
: 2240	3747	0	ELUDOM	

#### PSECT SUMMARY

Name	Bytes	Attributes
_COB\$CODE	2951	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

#### Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_S255\$DUA28:[SYSLIB]STARLET.L32;1	9776	27	0	581	00:00.8
_S255\$DUA28:[COBRTL.OBJ]SMGLIB.L32;1	469	0	0	38	00:00.2

#### COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD, INITIAL, OPTIMIZE)/NOTRACE/LIS=LIS\$:COBACCECV/OBJ=OBJ\$:COBACCECV MSRC\$:COBACCECV/UPDATE=(ENH\$:COBACCECV  
; )

; Size: 2929 code + 22 data bytes  
; Run Time: 00:42.5  
; Elapsed Time: 04:56.0  
; Lines/CPU Min: 5286  
; Lexemes/CPU-Mir: 21421



COBSACCECV      COBSACCECV - ACCEPT Conversion routines  
1-001            COBSSCAN\_INPUT - Scan the input string

<sup>K 1</sup>  
15-Sep-1984 23:49:06

VAX-11 Bliss-32 V4.0-742

Page 63

: Memory Used: 364 pages  
: Compilation Complete



0060 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

TYPMAIN  
LIS

COBPROLOG  
REQ

COBACCDAT  
LIS

COBACCDWK  
LIS

UNLOCK  
LIS

UTILSUBS  
LIS

INTPAR  
SDL

COBDEF  
REQ

COBACCDAY  
LIS

COBACCUCU  
LIS

COBRTL

COBLNK  
REQ

COBRTL  
MAP



0061 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY